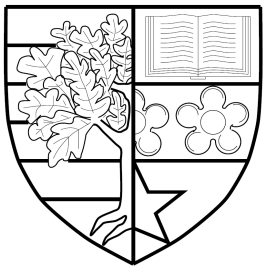
LOCAL TIME STEPPING METHODS AND DISCONTINUOUS GALERKIN METHODS APPLIED TO DIFFUSION ADVECTION REACTION EQUATIONS

by

Assionvi Hove Kouevi



Submitted for the degree of
Doctor of Philosophy

School of Mathematical and Computer Sciences
Heriot-Watt University

December 8, 2017

Abstract

Partial differential equations (PDEs), especially the diffusion advection and reaction equations (DAREs), are important tools in modeling complex phenomena, and they arise in many physics and engineering applications. Due to the difficulty of finding exact solutions, developing efficient numerical methods for simulating the solution of the DAREs is a very important and challenging research topic.

In this work, we present the transformation of the DAREs to ordinary differential equations (ODEs) using the standard finite element (FE) or the discontinuous Galerkin (DG) spatial discretization method. The resulting system of ODEs is then solved with standard time integrators such as implicit Euler methods, integrating factor method, exponential time differencing methods, exponential Rosenbrock methods, orthogonal Runge-Kutta Chebyshev methods. To illustrate the limitations of the FE method, we simulate and invert the cyclic voltammetry models using both spatial discretization methods (i.e. FE and DG) and show numerically that DG is more efficient.

In many physical applications, there are special features (such as fractures, walls, corners, obstacles or point loads) which globally, as well as locally, have important effects on the solution. In order to efficiently capture these, we propose two new numerical methods in which the mesh is locally refined in time and space. These new numerical methods are based on the combination of the DG method with local time stepping (LTS) approaches. We then apply these new numerical methods to investigate two physical problems (the cyclic voltammetry model and the transport of solute through porous media). These numerical investigations show that the combination of the DG with the LTS approaches are more efficient compared to the combination of DG with standard time integrators.

Acknowledgements

First and foremost, I would like to thank my supervisor Prof. Gabriel J. Lord and my co-supervisor Prof. Sebastian Geiger for their guidance, availability and great enthusiasm about research. The many discussions we had, were always illuminating. I would also like to sincerely thank Prof. Can Evren Yarman and Prof. Débora Campos de Faria, who provided and guided me during my internship at Schlumberger Gould Research in Cambridge. Part of this work was done during the productive periods of this internship. I am grateful for their indispensable motivation and career advises. I would like to express my deepest gratitude to Dr. Emma Spurlock who encouraged me to pursue my PhD study at Heriot Watt University and Prof Niel Turok for his foresight in founding African Institute for Mathematical Sciences (AIMS), my former institute, where I met Prof. Gabriel Lord and Dr. Emma Spurlock respectively as visiting lecturers and tutor. Many thanks also go all my lecturers at the University of Lome in Togo, especially Prof. Sena A. D'Almeida, Dr. Boussari Atanda, Dr. Edarh Bossou, Dr. Horatio Menouwagbe Quadjovie, for their guidance, support and profound knowledge throughout my undergraduate studies.

J'aimerais tout spéecialement remercier ma famille pour leur support inconditionnel et leurs encouragements incessants, depuis le tout début de mes études. J'ai également une pensée pour mes grands-parents, qui voient finalement aboutir mes longues années d'études. Mes amis ainsi que ma petite amie ont également eu un rôle très important: ils m'ont permis de décrocher des mathématiques et de me changer les idées à de nombreuses occasions. Merci à vous tous d'être là pour moi, même si je n'ai jamais vraiment réussi à vous expliquer le sujet de ma thèse!

Finally, I would like to thank Prof. Emmanuil Georgoulis and Prof. Lehel Banjai, respectively the external and internal examiners, for reading this document and submitting a report in a very short amount of time.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation and applications

An issue of incredible significance in natural science is to understand how substance or, on the other hand, organic contaminants are transported, for instance through a permeable media or amid an electrochemical response. It emerges in issues such as the transport of pollution, development of bacteria, oil and gas recovery from hydrocarbon supplies, contamination of groundwater tainting and manageable utilization of groundwater assets, putting away greenhouse gases (e.g, CO2) or radioactive waste in the subsurface, and mining heat from geothermal supplies. It is well known, see for example [159], that the movement of compound or organic contaminants can be depicted by the diffusion advection and reaction equations (DAREs), defined as follows

$$\frac{\partial C}{\partial t} + \nabla \cdot (\mathbf{v}C - D\nabla C) = f \ \text{ in } \ ]0,T] \times \Omega. \tag{1.1}$$

Here, $C(\mathbf{x},t)$, $\mathbf{x}$ in $\Omega \subset \mathbb{R}^n, t$ in $]0,T] \subset \mathbb{R}_+^*$ is the concentration of chemical or biological contaminants measured in mass per unit volume, D is the coefficient of diffusion measured in length squared per unit of time, $\mathbf{v} \in \mathbb{R}^n$ is the flow velocity measured in length per unit of time and f measured in mass per unit volume per unit of time, is the reaction term that model the generation or decay of the chemical or biological contaminants. In general f, D and $\mathbf{v}$ depend on space, time or the concentration itself. The DAREs are derived from the mass balance equation, which

states that the rate of change of the total mass in an arbitrary section of the medium must equal the net rate that mass flows into the section through its boundaries, plus the rate that mass is created, or destroyed, within the section. Due to the great mathematical difficulty of finding exact solutions, various numerical approaches have been developed.

A numerical method for DAREs can be developed, mainly by two ways. One follows the method of lines in which (1.1) is first semidiscretized in space, using the finite difference (FD), finite element (FE), finite volume (FV) or the discontinuous Galerkin (DG) methods, yielding a system of coupled ordinary differential equations (ODEs). The system of ODEs, obtained from the spatial discretization, would then be resolved with standard time integrators. Other can follow the space-time methods which consider the time direction like one of the spatial directions, and discretize (1.1) directly in both space and time by using FD, FE or DG methods. The DG method is successfully applied to DAREs and there is a large literature on the area, see for example, [98, 208, 133, 221, 18, 13, 57, 195, 116, 43]. We examine a new application in this area of cyclic voltammetry.

Since their introduction in 1973 by Reed and Hill [189] to recreate neutron transport, the DG methods have initiated broad enthusiasm attributable to the quantity of points of interest they offer in the numerical re-enactment. As indicated by the correlation of these spatial discretization techniques condensed in Tab 1.1, we can presume that DG technique consolidates attractive highlights of the FD, FE and FV methods.

	Complex geometries	High order accuracy	Conservation laws
FD	No	Yes	Yes
FV	Yes	No	Yes
FE	Yes	Yes	(No)
DG	Yes	Yes	Yes

Table 1.1: **Comparison of spatial discretisation method**: A "Yes" represents success, while "No" indicates a short-coming in the method. Finally, a "(No)" reflects that the method, with modifications, is capable of solving such problems but remains a less natural choice [125].

Beside the advantages stated in Tab 1.1, the increasing popularity of the DG method, is also due to other interesting numerical properties, stated as follows.

- Efficient computation properties: the discontinuities lead to a block-diagonal mass matrix, a diagonal matrix or identity matrix respectively if local basis polynomials are non orthogonal, orthogonal or orthonormal. The mass matrix can then be easily inverted. This is on account of the local elemental bases that can be selected freely because of the lack of any conformity requirement across the element interfaces. This infers there is no requirement for mass-lumping, utilized in the case of FE method, which can be problematic when using high-order finite element bases.

- Suitable for *hp*-adaptivity: mesh adaptivity is a noteworthy matter, particularly for hyperbolic equations, given the complexity of the solution structure. The DG method gives a simple and effective approach to bargain locally with mesh adaptation.

- Intended for advection equations: FE method is predominantly well suited for resolving diffusion equations, although advection-dominated equations necessitate the introduction of a sufficient quantity of numerical diffusion to stabilize the continuous methods. Using the DG method, the discontinuities between elements enable one to use high order advection schemes, for instance the approximate Riemann solvers, which makes the DG method extremely accurate for resolving advection-dominated equations.

However, the development and analysis of DG methods has followed two somewhat parallel routes depending on whether the PDE is hyperbolic or elliptic.

For hyperbolic PDEs, the first mathematical analysis of DG methods was performed by Lesaint and Raviart in 1974 [153, 152] and then upgreded by Johnson et al. [142] in 1984. All the more as of late, DG methods for hyperbolic and nearly hyperbolic equations encountered a noteworthy advancement in view of the thoughts of numerical fluxes, approximate Riemann solvers, and slope limiters; see for instance Cockburn et al. [60] and the references therein. For elliptic PDEs, the DG methods

started from the early work of Nitsche on boundary-penalty methods [173, 174] and the utilization of Interior Penalties (IP) to feebly enforce continuity on the solution or its derivatives over the interfaces among the connecting components; see for instance, Babuška [14], Babuška and Zlámal [15], Douglas and Dupont [86], Baker [17], and Wheeler [220]. Comparative reason was trailed by Arnold [10] formulating another finite element method for second-order parabolic equations where discontinuous piecewise polynomial functions were used over general meshes. By rewriting elliptic equations in mixed form, i.e. rewrite in the system of two first order equations, the DG methods were introduced more recently, see Bassi and Rebay [24] for the compressible Navier-Stokes equations, and further stretched out by Cockburn and Shu [63] for convection-diffusion problems, leading to the so-called Local Discontinuous Galerkin (LDG) method. The LDG has been effectively studied and applied to Stokes, Oseen and Navier-Stokes equations, refer Cockburn et al. [58]. However, one major drawback of LDG is the loss of compactness due to the introduction of lifting factors. Explicitly, the LDG stencil goes past immediate neighbours, in front of the usual DG stencil where degrees of freedom in one element are connected only to those in the neighbouring elements. To avoid this loss of compactness, compact discontinuous Galerkin (CDG) was introduced by Peraire and Persson [180] with application to elliptic problems. The CDG is fundamentally the same as LDG but it eliminates coupling between degrees of freedom of non-neighbouring elements by means of alternative local lifting operators, recovering the compactness lost with LDG. In the instance of the IP methods, unlike for the LDG or CDG, there is no need to write the problem as a first-order partial derivative equation and no additional variables or lifting operators have to be introduced. But the drawback of a DG formulation is that for the same mesh, its cost is, in general, higher than the one of a continuous formulation because of the duplication of the degrees of freedom at the element's boundaries.

The DG method is now widely used for solving a large variety of problems in many fields of practical engineering such as computational fluid dynamics [127, 190], aeroacoustics problems [53], magnetohydrodynamics or hydrodynamics [219] and

ocean modeling [191, 118, 117]. In many physical applications, there are special features (such as fractures, walls, corners, obstacles or point loads) which globally as well as locally, have an important effect on the solution. To capture these local behaviours, spatial local refinement is necessary, which can easily lead to a large scale system of ODEs when using the DG discretization method. Moreover, this requires a reduction of the time step, compared to the one used with a coarse mesh, in order to accurately simulate the solution in the refined zone and to avoid convergence problems when solving the discretized equations. Therefore, when applied uniformly on all the simulation domain, this reduced time step and the large scale DG discretized system of equations lead to an unacceptable computation time. Thus, the use of the so-called local time stepping (LTS) methods, which use the standard time integrator with different time steps on different regions of the solution domain, is highly desirable. Let us mention that the combination of the DG and LTS methods has been considered in the particular case of Maxwell's equations in [184] and wave equation in [91, 19]. It has also been used in [161] to solve unsteady heat conduction and diffusion problems in multi dimensions, where the DG discrete space-time variational formulation and an explicit approximative solution as predictor are used.

1.2 Objectives

The main goal of this thesis is the development of new efficient numerical methods based on the blending of the domain decomposition, DG and LTS techniques to solve DAREs and more particularly to investigate the cyclic voltammetry models, or flow and transport of solute through a porous media. To reach this objective, various partial objectives are accomplished:

1. derive the IP-DG formulation, providing non symmetric and coercive bilinear weak forms for the 1D cyclic voltammetry models (forward model),

2. derive the adjoint equation in order to provide the gradients for the inversion of the voltammetric signal (Inverse model),

3. develop a Matlab code, in order to invert synthetic data of the cyclic voltam-

metry models demonstrating the applicability of the proposed inverse model,

4. analyse and study the behaviour of the proposed forward and inverse models,

5. derive the IP-DG formulation for the 2D flow and transport of solute through in a domain with holes or fracture, assuming that the flow velocity is in line with Darcy's equation (forward model),

6. develop a Matlab code, in order to simulate the 2D flow and transport of solute through in a domain with holes or fracture, demonstrating the applicability of the IP-DG method,

7. propose two numerical methods (denote LTS-DG) for solving DAREs, combining the domain decomposition, DG and LTS methods to reduce the computational time of the two forward models introduced,

8. develop a Matlab code, in order to simulate 1D cyclic voltammetry models, or the 2D flow and transport of solute through in a domain with holes or fracture, demonstrating the applicability of the proposed LTS-DG methods,

Let us mention that the partial objectives, $1-4$, were achieved during my internship at Schlumberger Gould Research (SGR) centre in Cambridge, where previously a commercial available finite element algorithm was used. Thus the powerlessness of further improve performance of the inversion model, particularly the reduction of the computation time with the adjoint method (which is based on the operators obtained from the semidiscretization of the forward model).

1.3 Thesis Outline

This thesis is organized as follows:

In Chapter 2 we review the concepts of well-Posedness for Linear Model problems and the implementation of the time integrators of ODEs such as implicit Euler methods [44], Integrating factor method [149], exponential time differencing methods [131, 129], exponential Rosenbrock methods [132] and Orthogonal Runge-Kutta

Chebyshev methods [1]. Blending with the FE discretization, we numerically investigate the convergence of these time integrators when applied to the DAREs.

Chapter 3 introduces the DG analysis along with the implicit Euler method or exponential time differencing method, to simulate the one dimensional cyclic voltammetry models. To that end, we give a concise survey of the cyclic voltammetry and then apply the DG discretization (using Legendre polynomials) to their governing equation. In order to show the importance of the better conservation property of the DG method, we compare here its result with the one obtained by the MATLAB code "`pdepe`", which is based on the classic FE method [205] for space discretization and "$ode15s$" algorithm, described in [201, 202], as the time discretization.

Chapter 4 investigates the effect of the better conservation property of the DG method while fitting the data to extract the mechanistic information of the cyclic voltammetry models. For the seek of efficiency, the adjoint method described in [49, 213], is used for the computation of the gradient, instead of finite differences. We present several numerical experiments to propose a more efficient numerical inversion method.

Chapter 5 is devoted to the approximation of the model of flow and transport in porous media. To do so, we first review the concept of flow and transport in porous media. Secondly, we review the DG method for the spatial discretization of the DAREs in two and three dimensions. Finally, the presented DG method is combined with the mentioned standard time integrators, to simulate the model of flow and transport in a two dimensional domain with holes as well as a domain with a fracture.

Chapter 6 focus on reducing the computational time while simulating the cyclic voltammetry models, or flow and transport in porous media, using the DG method. To that end, we introduce two local time stepping methods based on domain decomposition techniques. For each novel solver, we present the results of several numerical experiments.

Chapter 7 concludes the thesis by summarizing the main contributions and major findings. We also suggest here some recommendations for future research.

In addition to the main chapters, we present in the appendix the details of the properties of the Legendre polynomials, the projection onto the finite space of the DG method and computation of several entities needed for the application of the DG and adjoint method.

Chapter 2

Finite element and time stepping methods

Contents

The goal, in this chapter, is to review how the continuous Galerkin method, commonly called the finite element (FE) method, can be combined with the standard time solvers (such as Euler, integrating factor, exponential differencing, exponential Rosenbrock and the orthogonal Runge-Kutta Chebyshev methods) to solve the DAREs given by (1.1). To that end, we first review in Section 2.1 the mathematical tools necessary to aplly the FE method. Secondly, we investigate in Section 2.2 the FE discretization for the DAREs in two dimensions based on piecewise linear functions. Thirdly, we review in Section 2.3 the standard time solvers mentioned above. Finally in Section 2.4, we present several numerical experiments. The novel contribution here is comparison of performance of FE combined with the mentioned time solvers for some two dimensions DAREs.

2.1 Basic concepts

The goal in this section is to recall the mathematical tools used for the application of the FE method. This is based on a variational formulation, which consists of seeking the solution of a boundary value problem in an appropriate functions space, e.g. H_0^1 for the Poisson's equation with Dirichlet boundary conditions [38]. Information on functional analysis can be found for example in [124, 80]. The variational problem is then reduced to a finite dimensional problem, which is the only one that can be handled by a computer, by seeking the solution in a finite dimensional subspace of the original function space. The variational formulation itself is either unmodified or slightly perturbed by a term which tends to zero at convergence. For this reason the mathematical tools for proving convergence are closely related to the tools for proving existence and uniqueness of the solution of the initial problem. More detailed description on the FE method can be found for example in the book by Brenner and Scott [38] or Ern and Guermond [97]. There is a large literature on FE method and some other good reference works are [225, 207, 79, 135, 56, 84, 25, 188].

The equation posed in (1.1), which imposes continuity and differentiability requirements on its potential solutions, is called the strong form. In general, the idea to obtain the weak form, also called the variational problem, is to first multiply the strong form by a smooth test function, then integrate over the whole domain and finally eliminate the highest derivatives with a possible integration by parts. By construction all solutions of the strong form satisfy the weak form but not vice-versa.

Example 2.1. Let us consider, for example, the Poisson equation i.e. $-\nabla^2 u = f$ on a domain $\Omega \subset \mathbb{R}^d$ with $f \in L^2(\Omega)$; subject to the Dirichlet boundary conditions i.e. $u = 0$ on the boundary $\partial\Omega$. Then, by the means of Green's identity [157] , we have for all test functions $v \in H_0^1(\Omega)$

$$F(v) = \int_\Omega fv = \int_\Omega -(\nabla^2 u)v = \int_\Omega \nabla u \cdot \nabla v = a(u, v). \tag{2.1}$$

Thus in this case $-\nabla^2 u = f$ is the strong form, while the weak form is given by

$$\text{Find } \; u \in U \text{ s.t. } a(u,v) = F(v) \;\; \forall v \in V, \tag{2.2}$$

where U and V, respectively called the trial and test space, are both equal to $H_0^1(\Omega)$. The weak form is then an alternate representation of the differential equation, which relaxes the requirements on solutions to a certain extent. This means that a larger set of functions are solutions of the weak form.

A standard strategy to prove the existence of solution to the strong form is to prove the existence of a function which satisfies the weak form (which is often easier to prove) and then proving that the function is sufficiently continuous and differentiable to satisfy the strong form. If $U = V$, the classical theorem for this is the Lax Milgram theorem that reads as follows

Theorem 2.1. *(Lax Milgram [80]) Let V be a Hilbert space. Suppose $a(\cdot,\cdot)$ is a continuous bilinear form on $V \times V$ and is coercive i.e.*

$$\exists \alpha > 0 \; s.t. \; \forall u \in V, \; a(u,u) \geq \alpha \parallel u \parallel_V^2 .$$

Let F be a continuous linear form on V, then the variational problem admits a unique solution and we have a priori estimate

$$\forall F \in V', \; \parallel u \parallel_V \leq \frac{1}{\alpha} \parallel F \parallel_{V'},$$

where V' is the dual space i.e. the space of linear functions on V.

It is often mandatory or more efficient to have the trial space U, where the solution is sought, to be different from the test space V in which the test function live i.e. $U \neq V$. The appropriate theoretical tool used in this case, is called Banach-Nečas-Babuška (BNB) theorem reads as follows

Theorem 2.2. *(Banach-Nečas-Babuška [80]) Let U be a Banach space and let V be a reflexive Banach space. Let us also assume that $a(\cdot,\cdot)$ is a continuous bilinear*

form on $U \times V$ and F is a continuous linear form on V and that the following two hypotheses are verified

1. *There exists $\alpha > 0$ such that*

$$\exists\, \alpha > 0, \quad \inf_{u \in U} \sup_{v \in V} \frac{a(u,v)}{\| u \|_U \| v \|_V} \geq \alpha.$$

2. *If $a(u,v) = 0, \forall v \in V$ then $u = 0$.*

Then the variational problem admits a unique solution and we have the priori estimate

$$\forall F \in V', \ \| u \|_U \leq \frac{1}{\alpha} \| F \|_{V'} .$$

The Lax Milgram theorem is a special case of BNB theorem. Indeed, if $U = V$ and the bilinear form $a(\cdot,\cdot)$ is coercive, then for any $v \in V$, we have

$$\alpha \| v \|_V \leq \frac{a(v,v)}{\| v \|_V} \leq \sup_{u \in V} \frac{a(u,v)}{\| v \|_V}. \tag{2.3}$$

Dividing by $\| v \|_V$ and taking the infimum gives condition 1 of BNB, the so called inf-sup condition. If $a(u,v) = 0, \forall v \in V$, then we have for $u = v$, $a(v,v) = 0$ and because of coercivity, this implies $v = 0$.

Let us now focus on the analysis of Galerkin discretization. The idea is simply to construct finite dimensional subspaces $U_h \subset U$, $V_h \subset V$ and to write the variational formulation replacing U and V respectively by U_h and V_h. Expanding the functions on bases of U_h and V_h yields a finite dimensional linear system that can be solved with standard methods. The convergence theorem for the Galerkin approximation reads as follows.

Theorem 2.3. *[92] Let U, U_h, V and V_h be Banach spaces and let $a(\cdot,\cdot)$ be a continuous bilinear form on $U \times V$, with continuity constant $\mathcal{C} > 0$ i.e. for all $w \in U$, $v \in V$ we have $| a(w,v) | \leq \mathcal{C} \| w \|_U \| v \|_V$. Let us assume that the exact solution $u \in U$ and the approximate solution $u_h \in U_h$ satisfy the Galerkin orthogonality condition i.e.*

$$a(u - u_h, v_h) = 0, \quad \forall v_h \in V_h,$$

and the bilinear form, $a(\cdot,\cdot)$, satisfies the discrete inf-sup condition for $\alpha > 0$

$$\alpha \parallel w_h \parallel_U \leq \sup_{v_h \in V_h} \frac{a(w_h, v_h)}{\parallel v_h \parallel_V}, \quad \forall w_h \in U_h. \tag{2.4}$$

Then the following error estimate holds

$$\parallel u - u_h \parallel_U \leq \left(1 + \frac{\mathcal{C}}{\alpha}\right) \inf_{w_h \in U_h} \parallel u - w_h \parallel_U .$$

The continuous Galerkin application is now reduced to finding a good approximation U_h of the trial space U which approximates well U. This can be done by choosing an approximation space based on piecewise polynomials of degree k. In this case, it can be proven that

$$\inf_{w_h \in U_h} \parallel u - w_h \parallel_{L^2} \leq c h^{k+1},$$

where h is related to the cell size. If the discrete inf-sup constant α does not depend on h, the error in FE approximation is the same, up to a constant, as the best approximation. The error is said to be optimal in this case, more details can be found for example in [97].

2.2 FE discretization for DAREs

The goal in this section is to discretize the DAREs, given by (1.1) by the FE method [65, 139, 113, 111]. We focus on the so-called method of lines in which (1.1) is first semidiscretized in space yielding a system of coupled ODEs which is then solved by time discretisation. To do so, we consider (1.1) on a domain $\Omega \subset \mathbb{R}^d$, subject to Dirichlet boundary condition (i.e. $C = 0$ on $\partial\Omega$), with diffusion coefficient $D(\mathbf{x}) \in C_0^\infty(\bar{\Omega})$, flow velocity $\mathbf{v}(\mathbf{x}) \in C_0^\infty(\bar{\Omega})$ and reaction term $f := f(C)$. The standard finite element method is applied, starting from the weak formulation. Once the weak form is obtained, we define the finite approximate space of the test and the trial space, using the linear piecewise polynomials.

2.2.1 FE discretization of DAREs

Let us first derive the weak form of the DAREs subject to the zero Dirichlet boundary condition. Multiplying (1.1) by a test function $v \in H_0^1(\Omega)$, which does not depend on the time t and integrating by means of the divergence theorem [157], yields the weak form

$$\text{Find } C \in H_0^1(\Omega),\ \int_\Omega \frac{\partial C}{\partial t} v - \int_\Omega (\mathbf{v}C - D\nabla C)\cdot\nabla v = \int_\Omega fv,\ \forall v \in H_0^1(\Omega). \quad (2.5)$$

Note here that the test and trial spaces are equal to $H_0^1(\Omega)$. Thus, the existence and the uniqueness can be proven by Theorem 2.1 for $U = V = H_0^1(\Omega)$. For the rest of this chapter, we assume that the problem we are working on is in two dimensions.

Now that the weak form has been derived, we focus on the finite approximate space of the original test and trial space V. Let us suppose that the domain $\Omega \in \mathbb{R}^2$ has a polygonal boundary $\partial\Omega$ and we can cover $\bar{\Omega}$ by a regular triangulation $\mathcal{T}$ of a closed triangles [181]. It means that the nodes of the mesh lie on the vertices of the triangles, the element of the triangulation do not overlap and no node lies on an edge of a triangle $T \in \mathcal{T}$. We introduce the notations $\mathcal{F}, \mathcal{F}_i$ and $\mathcal{F}_e$ to respectively denote the set of all edges, all internal edges and all external edges. This is illustrated in Fig 2.1, by plotting the domain $\Omega = [0,1] \times [0,1]$ partitioned into finite triangular elements using the function *distmesh2d* in MATLAB. For more details about the function *distmesh2d*, see [179, 178].

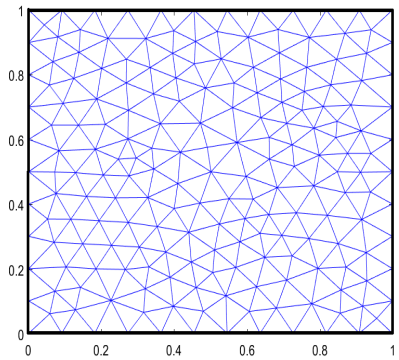


Figure 2.1: Domain $\Omega = [0,1] \times [0,1]$ partitioned into finite triangular elements using the function *distmesh2d* in MATLAB. The internal edges are in blue and the external edges are in black.

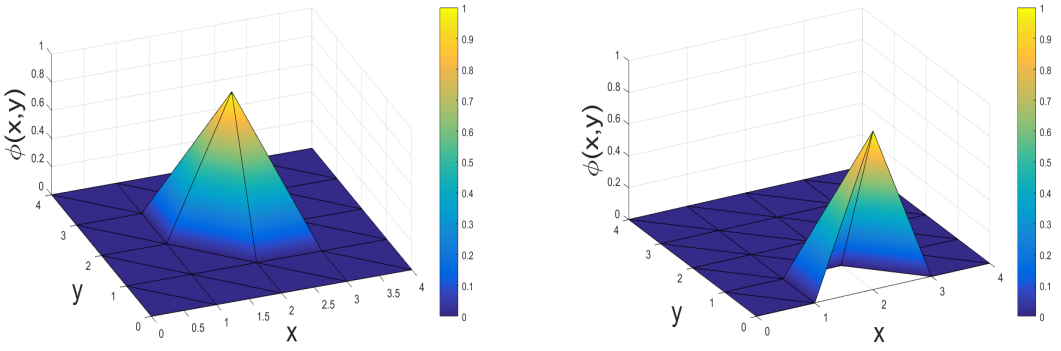
The good thing with the triangular elements is that they can approximate a huge range of geometries, while the quadratic elements have difficulties at corners. Each triangle $T \in \mathcal{T}$ is represented by three different nodes i.e. $T = (N_1^T,\ N_2^T,\ N_3^T)$. Each node $N_p = (x_p, y_p),\ p \in \{1, \cdots, \mathcal{N}\}$ is shared by a set S_p of the several triangles. Let us denote by r_p, the number of triangles in the set S_p, then we have

$$S_p = \left\{ T_j = (N_1^{T_j},\ N_2^{T_j},\ N_p),\ j = 1, \cdots, r_p \right\}.$$

Each node N_p is associated to one basis function ϕ_p such that $\text{supp}(\phi_p) = S_p$ and for all nodes $N_j = (x_j, y_j)$ we have $\phi_p(x_j, y_j) = \delta_p^j$. Let us assume here that ϕ_p is piecewise linear on each triangle $T_j \in S_p$ i.e. $\phi_p\,|_{T_j}\,(x, y) = a_p^{T_j} x + b_p^{T_j} y + c_p^{T_j}$. More explicitly, the restriction of the basis ϕ_p on the element $T_j \in S_p$ is given by

$$\phi_p\,|_{T_j}\,(x, y) = \frac{\mathcal{F}_p^{T_j}(x, y)}{\mathcal{F}_p^{T_j}(x_p, y_p)},\quad \mathcal{F}_p^{T_j}(x, y) = \det \begin{pmatrix} 1 & x & y \\ 1 & x_1^{T_j} & y_1^{T_j} \\ 1 & x_2^{T_j} & y_2^{T_j} \end{pmatrix}. \tag{2.6}$$

Therefore the approximate finite space $V_h = \text{ span}\,\{\phi_p,\ p = 1, \cdots, \mathcal{N}\}$. To illustrate these basis functions, we generate a uniform structured triangulation of the domain $\Omega = [0, 4] \times [0, 4]$, then we respectively plot in Fig 2.2(a) and Fig 2.2(b), the basis function associated to an internal node $A_i = (2, 2)$ and external node $A_e = (2, 0)$.



(a) (b)

Figure 2.2: **FE piecewise linear basis functions** : For a uniform structured triangulation of the domain $\Omega = [0, 4] \times [0, 4]$, we respectively plot in (a) and (b), the basis function associated to an internal node $A_i = (2, 2)$ and external node $A_e = (2, 0)$.

Now let us go back to the semi discretisation of (2.5). Using the definition of the approximate finite space V_h, we can approximate the solution C and the function f, at a given node (x, y) and time t, by

$$C(x, y, t) \approx \sum_{p=1}^{\mathcal{N}} X_p(t)\phi_p(x, y), \quad f(C) \approx \sum_{p=1}^{\mathcal{N}} f(X_p(t))\phi_p(x, y), \tag{2.7}$$

with $X_p(t) = C(x_p, y_p, t)$ for all $p = 1, \cdots \mathcal{N}$. By substituting (2.7) in (2.5) and taking the test function v as the basis functions ϕ_q, $q = 1, \cdots \mathcal{N}$, we obtain a system of ODEs

$$\mathbf{M}\frac{d}{dt}X - \mathbf{S}X = \mathbf{M}F_h(X), \tag{2.8}$$

where the vectors $X = [X_p(t)]$, $F_h(X) = [f(X_p(t))] \in \mathbb{R}^{\mathcal{N}\times 1}$ and by splitting the integral over the triangle, the entries of the matrices $\mathbf{M}$, $\mathbf{S} \in \mathbb{R}^{\mathcal{N}\times\mathcal{N}}$, so-called respectively mass and stiffness matrices, are given by

$$\mathbf{M}(p, q) = \sum_{T\in\mathcal{T}} \overbrace{\int_T \phi_p\phi_q d\mathbf{x}}^{m^T(\phi_p,\phi_q)}, \quad \mathbf{S}(p, q) = \sum_{T\in\mathcal{T}} \overbrace{\int_T (\mathbf{v}\phi_q - D\nabla\phi_q)\cdot\nabla\phi_p d\mathbf{x}}^{a^T(\phi_p,\phi_q)}, \tag{2.9}$$

for all $p, q \in \{1, 2, \cdots, \mathcal{N}\}$. Note from (2.9) that the mass matrix is symmetric.

2.3 Standard time stepping methods for ODEs

The main goal in this section is to review the standard time integrator that can be used to solve the system of ODEs (2.8) that I use in this thesis. To do so, let us split the function F_h into the sum of its linear and non linear parts (i.e. $F_h(X) = L_1X + N(X)$). Therefore, we can rewrite (2.8) as follows

$$\frac{d}{dt}X = LX + N(X), \quad L = \mathbf{M}^{-1}\mathbf{S} + L_1. \tag{2.10}$$

We now focus the time discretization of (2.10), since it is equivalent to (2.8).

2.3.1 Linearly-implicit method and implicit method

The Linearly Implicit (LI) method, often known as the semi implicit method, uses the backward Euler scheme for the linear term, and the forward Euler scheme for the non linear term. The LI method applied to (2.10), yields

$$(I - \Delta t L)X^{n+1} = X^n + \Delta t N^n, \tag{2.11}$$

where I is the identity matrix, Δt is the time step, X^n and N^n denotes respectively the numerical approximation of $X(t_n)$ and $N(X(t_n))$. Then starting from an initial condition X^0, the function X^n can be computed by repeatedly solving (2.11). The LI method is a first order method, and can extend to second order by using for example the trapezium rule for the linear term and the second-order Adams-Bashforth formula for the nonlinear terms, as described in [23, 44]. But we cannot extend beyond second order without losing $A-$stability (Dahlquist second stability barrier), see for example [70, 71] for more details.

The main idea behind the implicit (Impl) method is to first linearise (2.10) using the Taylor expansion, described in [170], for the non linear function N in each step at X^n. This leads to

$$\frac{\partial X}{\partial t} = \overbrace{\left(L + \frac{\partial N}{\partial X}(X^n)\right)}^{J_n} X + \overbrace{\left(N(X_n) - \frac{\partial N}{\partial X}(X^n)X_n\right)}^{N_n^n}, \tag{2.12}$$

where the matrices J_n and $\dfrac{\partial N}{\partial X}(X^n)$ are respectively the Jacobian of the function F_h and N at X^n. Finally, to complete the Impl scheme for (2.10), the backward Euler scheme is applied to the linear part of (2.12), to get

$$(I - \Delta t J_n)X^{n+1} = X^n + \Delta t N_n^n. \tag{2.13}$$

Then starting from an initial condition X^0, the function X^n can be computed by repeatedly solving (2.13). Note from (2.11) and (2.13) that LI and Impl are equivalent when (2.10) is linear i.e. $N = 0$.

2.3.2 First-order integrating factor Euler method

Integrating factor (IF) method appeared first in the work of Lawson [149], see for example [166] for a more comprehensive review. The basic idea of the IF method is to use the change of variable $u(t) = e^{-Lt}X(t)$ in (2.10), we obtain

$$\frac{u(t)}{dt} = e^{-tL}N(e^{Lt}u(t)). \tag{2.14}$$

The purpose now is to use any numerical integrator for (2.14) and then transform back to the approximated solution to the original solution X. For example, we can apply the Euler method [41] to (2.14) as follows

$$u^{n+1} = u^n + \Delta t e^{-t_n L}N(e^{Lt_n}u(t_n)),$$

where u^n is the numerical approximation of $u(t_n)$. This yields the first-order Integrating Factor Euler (IFEULER) method

$$X^{n+1} = e^{-\Delta tL}(X^n + \Delta tN^n). \tag{2.15}$$

The aim of transforming the differential equation (2.10) to equation (2.14), is to remove the explicit dependence in the differential equation on the operator L, except inside the exponential. The problem is no longer stiff since the linear term of the differential equation, that constrains the stability, is gone. Therefore, it can be solved exactly with the possibility of larger time steps. However, for PDEs with slow variation of the non linear terms, the introduction of the fast decay time scale into the non linear term introduces large errors into the system [30].

2.3.3 Exponential time differencing method

The exponential time differencing (ETD) method, see for example [131, 129] for more details, has been re-invented many times over the years, and unfortunately has been named differently [131]. The term "exponential time differencing" is used since this is how the method has been described in electrodynamics literature [131].

We multiply (2.10) by the integrating factor e^{-Lt} and integrate the result over a single time step, i.e. from $t = t_n$ to $t = t_n + \Delta t$, we obtain an exact formula

$$X^{n+1} = e^{\Delta t L} X^n + \int_0^{\Delta t} e^{(\Delta t - \tau)L} N(X(t_n + \tau)) d\tau. \tag{2.16}$$

This procedure does not introduce an unwanted fast time scale into the solution and the schemes can be generalized to arbitrary order. We consider the Taylor expansion of the non linear term N

$$N(X(t_n + \tau)) = \sum_{k=0}^{\infty} \frac{g_n^k}{n!} \tau^k, \quad g_n^k = \left. \frac{d^k}{dt^k} \right|_{t=t_n} N(X(t)). \tag{2.17}$$

By substituting (2.17) in (2.16) and using the change of variable $\theta = \tau / \Delta t$, we obtain

$$X^{n+1} = e^{\Delta t L} X^n + \sum_{k=1}^{\infty} \Delta t^k g_n^{k-1} \varphi_k(\Delta t L), \ \varphi_k(z) = \int_0^1 e^{(1-\theta)z} \frac{\theta^{k-1}}{(k-1)!} d\theta. \tag{2.18}$$

The ETD scheme is then built on the exponential function and the related functions φ_k, which can also be defined by the recurrence formula

$$\varphi_k(z) = \frac{\varphi_{k-1}(z) - \varphi_{k-1}(0)}{z}, \quad \varphi_0(z) = e^z.$$

The first order ETD1 scheme described in [48] and given by

$$X^{n+1} = \varphi_0(\Delta t L) X^n + \Delta t \varphi_1(\Delta t L) N^n, \tag{2.19}$$

is obtained by setting $k = 1$ in (2.18) or to impose that the function N is constant on $[t_n, t_n + \Delta t]$ (i.e. for all $t \in [t_n, t_n + \Delta t]$, $N(t) = N^n$). The second order ETD2 scheme described in [68] and given by

$$X^{n+1} = \varphi_0(\Delta t L) X^n + \Delta t \varphi_1(\Delta t L) N^n + \Delta t^2 \varphi_2(\Delta t L) \frac{N^n - N^{n-1}}{\Delta t}. \tag{2.20}$$

This is obtained by setting $k = 2$ in (2.18) or using the approximation

$$N(X(t)) \approx N^n + (t - t_n)\frac{N^n - N^{n-1}}{\Delta t},$$

for all $t \in [t_n, t_n + \Delta t]$. Note from (2.20) that the computation of X^1 requires the information about N^{-1}, which is not defined. Thus to implement ETD2, we assume that $N^{-1} = N^0$. In general, for the multi-step time-discretization methods, all the information required to start the integration is not available. Therefore, it is preferable to construct ETD methods based on the Runge-Kutta methods. The second order ETD Runge-Kutta (ETD2RK1) scheme has been described in [68, 130] by considering the approximation of the function N for all $t \in [t_n, t_n + \Delta t]$ as follows

$$N(X(t)) \approx N^n + (t - t_n)\frac{N(a_n) - N^n}{\Delta t}, \quad a_n = \varphi_0(\Delta t L)X^n + \Delta t\varphi_1(\Delta t L)N^n. \tag{2.21}$$

By substituting (2.21) into (2.16), we obtain the ETD2RK1 scheme

$$X^{n+1} = a_n + \Delta t^2\varphi_2(\Delta t L)\frac{N(a_n) - N^n}{\Delta t}. \tag{2.22}$$

It has been shown that, see [12], when solving stiff problems, such as (2.10), the selection of the time step size for these methods is only limited by accuracy and not the stability. It indicates the possibility of using a large time step and consequently ETD and ETDRK methods provide computational savings over conventional explicit methods.

2.3.4 Exponential Rosenbrock method

The main idea of the exponential Rosenbrock methods, reviewed in [132], is to linearise (2.10) in each step at X^n to get (2.12), which is then multiplied by the integrating factor $e^{-J_n t}$ and the result integrated over a single time step. This yields the second order Exponential Rosenbrock-Euler scheme

$$X^{n+1} = \varphi_0(\Delta t J_n)X^n + \Delta t\varphi_1(\Delta t J_n)N_n^n. \tag{2.23}$$

More generally, the class of s-stage exponential Rosenbrock-type schemes takes the following form

$$X^{ni} = e^{c_i h_n J_n} X^n + h_n \sum_{j=1}^{i-1} a_{ij}(h_n J_n) D_{nj},\ X^{n+1} = e^{h_n J_n} X^n + h_n \sum_{i=1}^{s} b_i(h_n J_n) D_{ni},$$

where X^{ni} is the solution at time $t_n + c_i h_n$, the weights $b_i(z)$ and the coefficients $a_{ij}(z)$ will be chosen such that

$$\sum_{i=1}^{s} b_i(z) = \varphi_1(z), \quad \sum_{j=1}^{i-1} a_{ij}(z) = c_i \varphi_1(c_i z), \quad 1 \le i \le s. \tag{2.24}$$

Thus, s-stage exponential Rosenbrock-type scheme require the definition of the coefficients c_i and the functions a_{ij}, b_i. Note that the equations (2.24) implies $c_1 = 0$ and consequently $X^{n1} = U_n$. This method can be reformulated as follows

$$\begin{aligned} N_n(X^{ni}) &= N_n(X^n) + D_{ni}, \quad 2 \le j \le s, \\ X^{ni} &= X^n + c_i h_n \varphi_1(c_i h_n J_n) F(X^n) + h_n \sum_{j=2}^{i-1} a_{ij}(h_n J_n) D_{nj}, \\ X^{n+1} &= X^n + c_i h_n \varphi_1(c_i h_n J_n) F(X^n) + h_n \sum_{i=2}^{s} b_i(h_n J_n) D_{ni}. \end{aligned}$$

In particular, let us consider the case $s = 2$. The weights b_i and the coefficients a_{ij} of the 2-stage exponential Rosenbrock-type method (EXPR) are defined by

$$\begin{array}{c|cc} c_1 & & \\ c_2 & a_{21} & \\ \hline & b_1 & b_2 \end{array} = \begin{array}{c|cc} 0 & & \\ 1 & \varphi_1 & \\ \hline & \varphi_1 - 2\varphi_3 & 2\varphi_3 \end{array}.$$

Therefore the 2-stage exponential Rosenbrock-type method can be reduced as follows

$$\begin{aligned} X^{n2} &= X^n + h_n \varphi_1(h_n * J_n) F(X^n), \\ D_{n2} &= N_n(X^{n2}) - N_n(X^n), \\ X^{n+1} &= X^{n2} + 2h_n \varphi_3(h_n * J_n) D_{n2}. \end{aligned}$$

Remark 2.1. The key element in the implementation of exponential integrators

schemes (IF, ETD and EXPR) is the computation of $\varphi_i(z)x$ for a given matrix z, a vector x and a matrix exponential function $\varphi_i,\ i = 0, 1, \cdots$. This can be done with the Padé approximation [168, 204], real fast Léja points technique [16, 29, 47] or the Krylov subspace technique [156, 128, 204]. It is well known that a standard Padé approximation for a matrix exponential functions is not an efficient method for large scale problems [204]. So we consider here the Krylov subspace or real fast Léja points technique, to implement ETD and EXPR.

Also note that if $N = 0$, then (2.10) is linear. In this case, the time solvers IF, ETD1, ETD2, ETDRK and EXPR are equivalent; and for a given initial time t_0 and final time t_1, we have $X(t_1) = e^{-(t_1-t_0)L}X(t_0)$, independently of the time step Δt used for the mentioned time solvers. This implies that the error between the exact solution and the solution simulated with IF, ETD1, ETD2, ETDRK or EXPR is independent of the time step.

2.3.5 Orthogonal Runge-Kutta Chebyshev methods

The orthogonal Runge-Kutta Chebyshev methods (ROCK) are obtained from a combination of the approach of Van der Houwen and Sommeijer (RKC) proposed in [216] and the approach proposed by Lebedev (DUMKA) in [151, 150]. These methods possess nearly optimal stability polynomials which are built on a recurrence relation. The basic idea of these methods is to search, for a given value of p, the approximation

$$R_s(z) = w_p(z)P_{s-p}(z) = 1 + z + \cdots + \frac{z^p}{p!} + \mathcal{O}(z^{p+1}), \tag{2.25}$$

where P_{s-p} is a member of the family of the orthogonal polynomials $\{P_j\}_{j\geq 0}$ with respect to the weight function $\dfrac{w_p(z)^2}{\sqrt{1-z^2}}$ and w_p is a positive polynomial of degree p.

Consider the family of polynomials $\{P_j\}_{j\geq 0}$, which satisfies (2.25) for $p = 2$. In this case, the ROCK scheme is denoted ROCK2. To compute X^{n+1}, the solution of (2.10), at the time t_{n+1}; the three-term recurrence formula associated with the family polynomials $\{P_j\}_{j\geq 0}$, i.e. $P_j(z) = (\alpha_j z - \beta_j)P_{j-1}(z) - \gamma_j P_{j-2}(z)$, is used to

define the internal stages of the method

$$\begin{aligned} K_1 =& X^n + \alpha_1 h F(X^n), \\ K_j =& \alpha_j h F(K_{j-1}) - \beta_j K_{j-1} - \gamma_j K_{j-2}, \quad j = 2, \cdots, s-2, \end{aligned}$$

where $K_0 = X^n$ and $h = \Delta t$. Then the quadratic factor $w_2(z) = 1 + 2\sigma z + \tau z^2$ is represented by the two-stage finishing procedure

$$\begin{aligned} K^*_{s-1} &= K_{s-2} + \sigma h F(K_{s-2}), \\ K^*_s &= K^*_{s-1} + \sigma h F(K^*_{s-1}), \\ X^{n+1} &= K_s - h\sigma(1 - \tau\sigma^{-2})(F(K^*_{s-1}) - F(K_{s-2})). \end{aligned}$$

In general, note that the ROCK method is also based on the computation of several coefficients $\alpha_i, \beta_i, \gamma_i, \sigma$ and τ. For the numerical experiments, we implement in MATLAB the ROCK2 scheme with a fixed stage $s = 4$. For details on this method, see for example [1], where Abdulle and Vilmart present the adaptive stage ROCK scheme.

2.4 Numerical experiments

The goal in this section is to first validate the FE method and its implementation investigated here for the DAREs, through several numerical experiments in two dimensions. Finally, we compare the performance of the FE method combined with the time solvers, such as LI, Impl, ETD1, ETD2, EXPR and ROCK2, to solve DAREs. To that end, we demonstrate the decay of the errors between the numerical and exact solutions at the final time as we either increase the dimension $\mathcal{N}$ of the finite space (convergence in space) or reduce the time step Δt of the time solver (convergence in time).

Convergence in space is implemented by using the uniform refinement of uniform mesh of the rectangular domain Ω. To construct the uniform mesh, for a given value $N_s \in \mathbb{N}$, we subdivide the domain $\Omega = [x_0, x_1] \times [y_0, y_1]$ into N_s^2 rectangles by

subdividing the interval $[x_0, x_1]$ and $[y_0, y_1]$ into N_s uniform intervals respectively in x and y directions. Then, each rectangle is split into two triangles. This construction is illustrated in Fig 2.3 for $N_s = 4$. So the dimension of the finite space of the FE method is $\mathcal{N} = (N_s + 1)^2$ and increases with N_s.

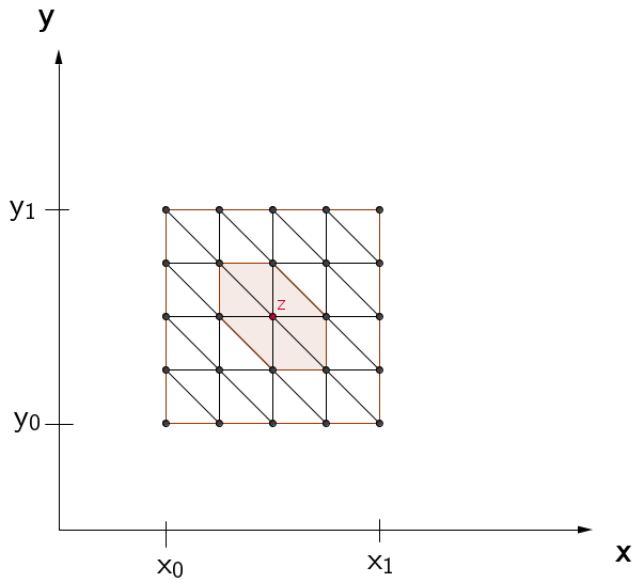


Figure 2.3: Uniform mesh of the domain Ω for $N_s = 4$ for FE method. We also highlight the support of the basis function associated to the node Z.

The convergence in time is examined by using the uniform refinement of uniform mesh of set of the time $[t_0, t_1]$, where t_0 and t_1 are respectively the initial and final time. The time step is given by $\Delta t = (t_1 - t_0)N_t^{-1}$, for a given $N_t \in \mathbb{N}$. Thus, DAREs can be solved once N_s and N_t are given.

2.4.1 Diffusion and advection without reaction

Let us consider the two-dimensional diffusion advection (DA) equation, by setting

$$\mathbf{v} = (u_0 x, -u_0 y)\,, \quad D = \begin{pmatrix} D_0 u_0^2 x^2 & 0 \\ 0 & D_0 u_0^2 y^2 \end{pmatrix}, \quad f(C) = 0, \tag{2.26}$$

in (1.1) for a given $u_0, D_0 \in \mathbb{R}$. Zoppou and Knight have shown in [226] that for a unit instantaneous release at (x_0, y_0), the evolution in time of the concentration

profile of this diffusion and advection equation is given by

$$C_e(x,y,t) = \frac{(xyx_0y_0)^{-\frac{1}{2}}}{4\pi D_0 u_0^2 t}\left(\frac{xy_0}{x_0y}\right)^{(2u_0D_0)^{-1}} \exp\left(\frac{-\phi^2 - 2(1+D_0^2u_0^2)t^2}{4D_0t}\right), \quad (2.27)$$

where $\phi = u_0^{-1}\sqrt{\ln^2(x/x_0) + \ln^2(y/y_0)}$. Note from (2.27) that the exact solution of (1.1) with the settings (2.26) and a unit instantaneous release at (x_0, y_0) is not defined at $t = 0$, $x = 0$ and $y = 0$ and follows

$$\lim_{x\to 0} C_e = \lim_{y\to 0} C_e = \lim_{x\to+\infty} C_e = \lim_{y\to+\infty} C_e = 0.$$

Thus, to simulate (2.27) at the time $t_1 = 0.1$ for $u_0 = 1, D_0 = 2, x_0 = y_0 = 5$, we solve (1.1) with the settings (2.26) on $\Omega = [0.01, 50] \times [0.01, 50]$ and subject to the Dirichlet boundary condition $C = 0$ on $\partial\Omega$ and the initial condition $C(t = 0.01) = C_e(t = 0.01)$ on Ω, by means of FE method combined with LI and ETD. So the initial time is given by $t_0 = 0.01$. Here we implement two types of ETD1, denoted ETD Leja and ETD Arnoldi respectively based on real fast Léja points and the Krylov subspace technique.

To investigate the decay of the errors between the numerical and exact solutions at the final time as we increase the dimension $\mathcal{N}$ of the finite element space, we simulate (2.27) at the time $t_1 = 0.1$ using FE combined with the Impl, ETD Leja, ETD Arnoldi for all $(N_s, N_t)_i,\ i \in \{1, \cdots, 5\}$ illustrated in Tab 2.1.

i	1	2	3	4	5
N_s	100	200	300	400	500
N_t	100				

Table 2.1: Settings to investigate convergence in space for DA.

We denote by C_i^r, the simulated concentration from the FE method combined with the time solver $r \in \{$LI, ETD Leja, ETD Arnoldi$\}$ for the setting $(N_s, N_t)_i,\ i \in \{1, \cdots, 5\}$. For each simulated solution C_i^r, we compute the error as follows

$$\text{error}_i^r = \| C_e - C_i^r \|_{L^2(\Omega)} \quad (2.28)$$

Throughout the simulation of C_i^r for all setting $(N_s, N_t)_i,\ i \in \{1, \cdots, 5\}$ and all time solver $r \in \{\text{LI, ETD Leja, ETD Arnoldi}\}$, we record the CPU time, CPU_i^r.

We respectively plot in Fig 2.4(a), Fig 2.4(b), Fig 2.4(c), the simulated (2.27) at the time $t_1 = 0.1$ obtained by using FE combined with the Impl, ETD Leja and ETD Arnoldi for the setting $(N_s, N_t)_1$ i.e. $N = N_t = 100$. For each solver used, we plot in Fig 2.4(d) the logarithm of the error $\log(\text{error}_i^r)$ against the logarithm of the dimension of the finite space $\mathcal{N}_i = (N_{s,i} + 1)^2$. As expected, Fig 2.4(d) shows the decay of $\log(\text{error}_i^r)$ as we increase the value of $\mathcal{N}_i$. It also shows that, for a given setting (N_s, N_t), ETD based on either real fast Léja points or Krylov subspace technique simulate the same solution which is more accurate compared to the one obtained Impl i.e.

$$\text{error}_i^{\text{ETD Arnoldi}} \approx \text{error}_i^{\text{ETD Leja}} < \text{error}_i^{\text{LI}}.$$

We plot in Fig 2.4(e), the logarithm of the error, $\log(\text{error}_i^r)$, against the logarithm of the CPU time, $\log(\text{CPU}_i^r)$. Note from Fig 2.4(e) that the simulated concentration with a given error, error_0, is obtained practically with the same CPU time while using FE method with ETD Leja or ETD Arnoldi. Moreover, it shows that

$$\text{CPU}_0^{\text{ETD Arnoldi}} \approx \text{CPU}_0^{\text{ETD Leja}} < \text{CPU}_0^{\text{LI}},$$

where CPU_0^r is the time spent, to simulate the concentration with the error error_0 ,by the FE method combined with the time solver $r \in \{\text{LI, ETD Leja, ETD Arnoldi}\}$.

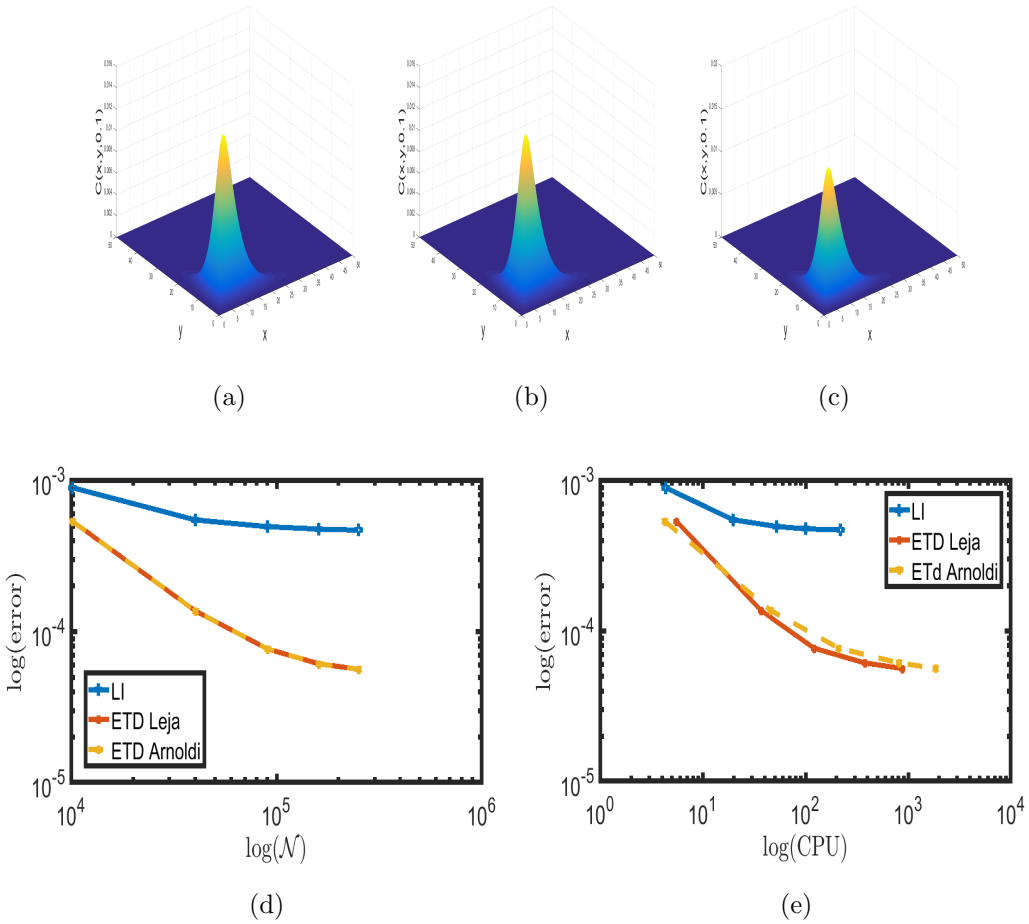


Figure 2.4: **Convergence in space for DA using FE method combined with LI, ETD Leja and ETD Arnoldi.** We respectively plot in (a), (b) and (c), the simulited concentration at the time $t_1 = 0.1$ obtained by using FE combined with the LI, ETD Leja and ETD Arnoldi for the setting $N = N_t = 100$. We plot in (d) and (e) the logarithm of the error, $\log(\text{error}_i^r)$, associated to simulated solution for setting $(N_s, N_t)_i$ respectively against the logarithm of the dimension of the finite space , $\mathcal{N}_i = (N_{s,i} + 1)^2$, and the logarithm of the CPU time, $\log(\text{CPU}_i^r)$. As expected, (d) shows that error_i^r decrease as we increase $\mathcal{N}_i$. Note from (e) that ETD here is more efficient compared to LI.

Let us now investigate the convergence in time in this case. To do so, we simulate (2.27) at the time $t_1 = 0.1$ using FE combined with the Impl, ETD Leja, ETD Arnoldi for all $(N_s, N_t)_i$, $i \in \{1, \cdots, 5\}$ illustrated in Tab 2.2.

i	1	2	3	4
N_t	30	60	100	200
N_s	500			

Table 2.2: Settings to investigate convergence in time for DA.

We follow the same procedure as previously to compute the error, error_i^r, associated to the simulated concentration, C_i^r, at the time $t_1 = 0.1$ for all setting $(N_s, N_t)_i, i \in \{1, \cdots, 4\}$ and all time solver $r \in \{\text{Impl, ETD Leja, ETD Arnoldi}\}$. We then plot in Fig 2.5 the logarithm of the error $\log(\text{error}_i^r)$ against the logarithm of the time step $\Delta t_i = (t_1 - t_0)N_{t,i}^{-1}$. As expected, we see from Fig 2.5 that the error is independent of the time step when we use ETD method, while it decreases with the time step when we use the Impl method. It also shows that ETD Leja and ETD Arnoldi lead to the same concentration which is more accurate compared to the concentration obtained using Impl i.e. for all time steps

$$\text{error}_i^{\text{ETD Arnoldi}} \approx \text{error}_i^{\text{ETD Leja}} < \text{error}_i^{\text{Impl}}.$$

Thus, we choose to implement the ETD and EXPR methods with the Krylov subspace technique from now on.

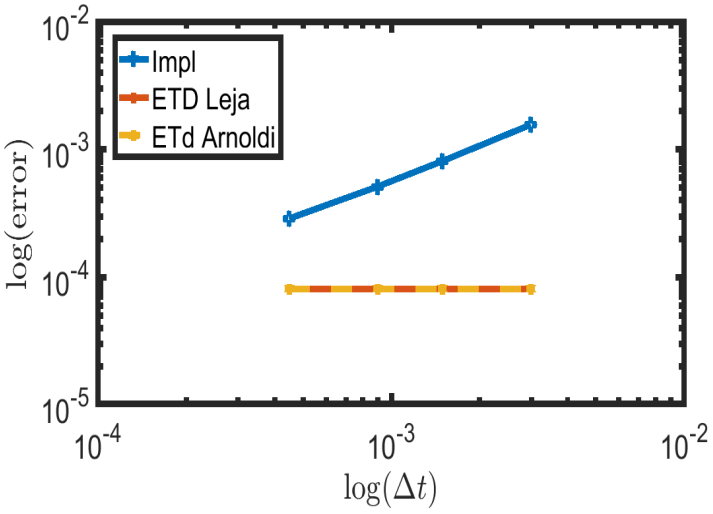


Figure 2.5: **Convergence in time for DA using FE method combined with LI, ETD Leja and ETD Arnoldi.** We plot in this figure, the logarithm of the error $\log(\text{error}_i^r)$ against the logarithm of the time step Δt_i. As expected, it shows that the error is independent from the time step when we use ETD method, while it is decreasing with the time step when we use Impl method.

2.4.2 Diffusion advection with a non linear reaction

Let us consider (1.1) with the settings (2.26) used in Section 2.4.1. But in this case, we assume that the function f is non linear and given by $f(C) = C - C^3$. We

then simulate the concentration profile at the time $t_1 = 0.1$ using the FE method combined with the time solvers LI, ETD1, ETD2RK1 and EXPR for the settings $(N_s, N_t)_i,\ i \in \{1, \cdots, 5\}$ as illustrated in Tab 2.3.

i	1	2	3	4	5
N_t	10	10^2	10^3	10^4	10^5
N_s	200				

Table 2.3: Settings to investigate convergence in time for DAREs.

Since the exact solution is unknown in this case, to demonstrate the decay of the error as we decrease the time step Δt, we assume that the exact solution is given by the finest time step i.e. Δt_5. The error between the exact and the simulated concentration is then computed as follows

$$\text{error}_i^r = \| C_5^r - C_i^r \|_{L^2(\Omega)},\ \forall i \in \{1, 2, 3, 4\}, \tag{2.29}$$

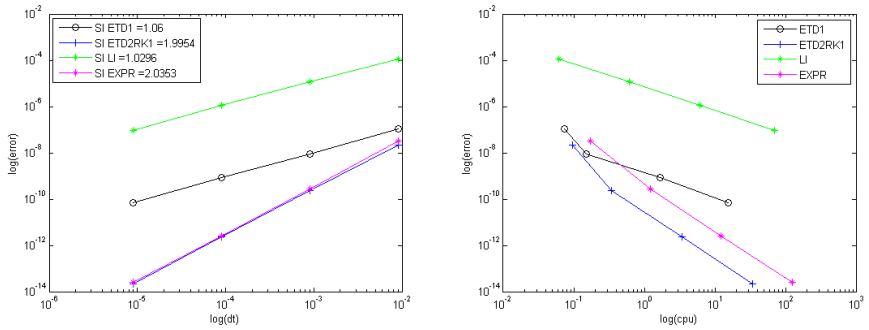
where C_i^r is the concentration simulated using the FE method combined with the time solver $r \in \{\text{LI}, \text{ETD1}, \text{ETD2RK1 and EXPR}\}$ for the setting $(N_s, N_t)_i,\ i \in \{1, \cdots, 5\}$. Throughout each simulation, we also record the CPU time, CPU_i^r.

We plot in Fig 2.6(a) and Fig 2.6(b) the logarithm of the error $\log(\text{error}_i^r)$ respectively against logarithm of the time step $\log(\Delta t_i)$ and the logarithm of the CPU time, $\log(\text{CPU}_i^r)$ for all $i \in \{1, 2, 3, 4\},\ r \in \{\text{LI}, \text{ETD1}, \text{ETD2RK1 and EXPR}\}$. Every data set $(\log(\Delta t_i), \log(\text{error}_i^r)),\ i \in \{1, \cdots, 4\}$ of the time integrator $r \in \{\text{LI}, \text{ETD1}, \text{ETD2RK1 and EXPR}\}$ can be approximated with a straight line, see [143], with a slope given by

$$Sl^r = \frac{\text{Cov}\left[\log(\Delta t_i), \log(\text{error}_i^r)\right]}{\text{Var}\left[\log(\Delta t_i)\right]}.$$

We the display in Fig 2.6(a), the slope Sl^r for all time integrators used. The value of the slopes represent the order of convergence; as expected, LI and ETD1 are all first order while ETD2RK1 and EXPR are second order. Note also from Fig 2.6(a) that the error error_i^r decrease with the time step Δt_i. Fig 2.6(b) shows that in the

increasing order of efficiency, we have LI<ETD1<EXPR<ETD2RK1.



(a) (b)

Figure 2.6: **Convergence in time for DAREs using FE method combined with LI, ETD1, EXPR and ETD2RK1.** We plot in (a) and (b) the logarithm of the the error $\log(\text{error}_i^r)$ respectively against logarith of the the time step $\log(\Delta t_i)$ and the logarithm of the the CPU time, $\log(\text{CPU}_i^r)$ for all $i \in \{1, 2, 3, 4\}$, $r \in \{\text{LI}, \text{ETD1}, \text{ETD2RK1 and EXPR}\}$. (a) that the error error_i^r decrease with the time step Δt_i while (b) shows that in the increasing order of efficiency, we have LI $<$ ETD1 $<$ EXPR $<$ ETD2RK1.

2.4.3 Diffusion with non linear reaction

Let us consider (1.1) on a domain $\Omega = [0, 1] \times [0, 1]$, with diffusion coefficient $D = 10^{-2}$, flow velocity $\mathbf{v} = (0, 0)$, and the reaction term $f(C) = C - C^3$. This equation, so-called diffusion and reaction (DR) equation, is then subject to Dirichlet boundary condition (i.e. $C = 0$ on $\partial\Omega$) and the initial condition $C(x, y, t = t_0 = 0) = \sin(2\pi x)\sin(\pi y)$. In this case, to investigate the convergence in time, we simulate the concentration profile at the time $t_1 = 1$ using FE method combined the time solvers such as LI, ETD1, ETD2 EXPR and ROCK2 for the all setting $(N_s, N_t)_i$, $i \in \{1, \cdots, 6\}$ illustrated in Tab 2.4.

i	1	2	3	4	5	6
N_t	10	10^2	5×10^2	10^3	10^4	5×10^4
N_s	100					

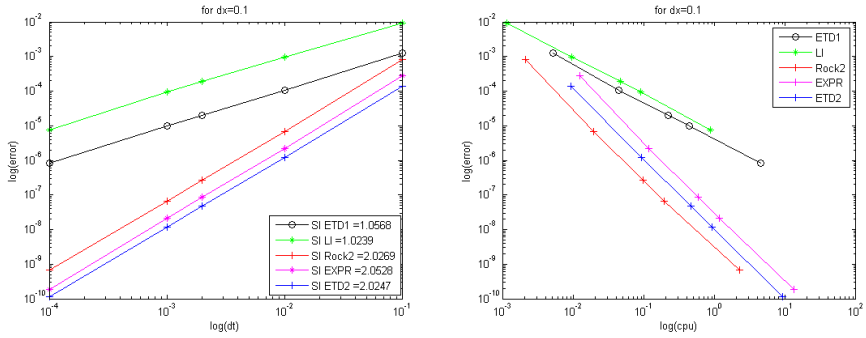
Table 2.4: Settings to investigate convergence in time for DR.

Once again, we assume that the exact solution here is also given by the simulated concentration for the finest time step. The error between the exact and the simulated

concentration is then given by

$$\text{error}_i^r = \| C_6^r - C_i^r \|_{L^2(\Omega)}, \ \forall i \in \{1, \cdots, 5\}, \tag{2.30}$$

where C_i^r is the concentration simulated using the FE method combined with the time solver $r \in \{\text{LI}, \text{ETD1}, \text{ETD2}, \text{EXPR and ROCK2}\}$ for the setting $(N_s, N_t)_i$, $i \in \{1, \cdots, 5\}$. We plot in Fig 2.7(a) and Fig 2.7(b) the logarithm of the the error $\log(\text{error}_i^r)$ respectively against logarithm of the the time step $\log(\Delta t_i)$ and the logarithm of the the CPU time, $\log(\text{CPU}_i^r)$ for all $i \in \{1, \cdots, 5\}$, $r \in \{$ LI, ETD1, ETD2 ,EXPR and ROCK2 $\}$. Fig 2.7(a) shows that the error error_i^r decrease with the time step Δt_i while Fig 2.7(b) shows that in the increasing order of efficiency, we have LI<ETD1<EXPR<ETD2<ROCK2. Fig 2.7(a) also shows that EXPR, ETD2, ROCK2 are second order but LI, ETD1 are first order, as expected.



(a) (b)

Figure 2.7: **Convergence in time for DR using FE method combined with LI, ETD1, ETD2 EXPR and ROCK2.** We plot in (a) and (b) the logarithm of the the error $\log(\text{error}_i^r)$ respectively against logarith of the the time step $\log(\Delta t_i)$ and the logarithm of the the CPU time, $\log(\text{CPU}_i^r)$ for all $i \in \{1, \cdots, 5\}$, $r \in \{\text{LI}, \text{ETD1}, \text{ETD2}, \text{EXPR and ROCK2}\}$. (a) shows that the error error_i^r decrease with the time step Δt_i while (b) shows that in the increasing order of efficiency, we have LI < ETD1 < EXPR < ETD2 < ROCK2.

2.5 Summary

In this chapter, we reviewed numerical method based on the FE space discretization and standard time discretization for the DAREs. To that end, we follow the description on the FE found for example in the book by Brenner and Scott [38] or

Ern and Guermond [97]. In two dimensions, we have applied the combination of the FE discretization with the time integrators such that LI, Impl, ETD, EXPR and ROCK2 method to a variety of linear and non-linear DAREs. However, the main goal of this chapter is to describe the time integrators methods that will be used in this thesis.

Chapter 3

One dimensional discontinuous Galerkin method for Cyclic Voltammetry models

Contents

The biggest challenge in the study of the cyclic voltammetry models is to find the analytical solution of their governing equations. So the electro-chemists rely on the software package or finite difference methods to solve numerically these equations [33, 36, 34, 35]. But this approach is strongly inefficient while inverting the cyclic voltammetry models by fitting its signal response. Therefore in this chapter, we proposed a more efficient numerical method, to solve the governing equations of the cyclic voltammetry models. In order to do so, we first give a brief review on cyclic voltammetry in Section 3.1, then we investigate the numerical resolution, using DG space discretization, of two cyclic voltammetry problems respectively in Section 3.2 and Section 3.3.

The contribution here is the investigation of the cyclic voltammetry, based on the

novel combination of the DG space discretization and standard time discretization method. The DG method relies on the Legendre polynomials while the standard time discretization method used are Impl and ETD method. The DG analysis presented here, can be used for any diffusion and second order non linear reaction term: it has not been applied in this context before. Contrary to expectations, while combine with the DG spatial discretization, the standard implicit time integrator out performs the methods such as the ETD method and adaptive time stepping method $ode15s$. We also see that the DG method performs better than the standard FE method.

3.1 Introduction to cycle voltammetry

Cyclic voltammetry (CV) is a technique used to study electrochemical reaction mechanisms that give rise to electroanalytical current signals. It has been frequently used by electrochemists for a variety of purposes, including chemical and biochemical sensing (e.g. glucose sensors [187], gas detectors [3, 200], pH meters [224]), technological applications (e.g. electroplating [224], electrochromic displays [28, 192]), energy storage (e.g. solar cells [223], batteries [146]), imaging, synthesis, which underpin much of modern biology and nanotechnology [140, 158, 138]. There are several good texts that investigate the theory and practice of CV in depth, see for example [22, 163].

3.1.1 Cycle voltammetry experiment

CV involves applying a voltage to an electrode immersed in an electrolyte solution, and seeing how the system responds. Let us consider for example the electron transfer only process at the electrode represent as follows

$$\mathbf{Q} \underset{k_b}{\overset{k_f}{\rightleftharpoons}} \mathbf{Q}^{n+} + ne^-, \tag{3.1}$$

where n is the number of electrons transferred per molecule, the rate constants k_b and k_f are given by the Buttle-Volmer kinetics Equation [66]. We have

$$k_b = k_0 \exp\left[(1-\alpha)\frac{nF}{RT}(E-E^0)\right],\ k_f = k_0 \exp\left[(-\alpha)\frac{nF}{RT}(E-E^0)\right], \qquad (3.2)$$

where $E(V)$ is the potential applied to the electrode, F is Faraday constant, $T(^\circ K)$ is the Kelvin temperature, R is the universal gas constant, $\alpha \in [0,1]$ is the charge transfer coefficient, $k_0(ms^{-1})$ is the the value of the rate constants at the formal potential $E^0(V)$. These rate constants describe how the flux of electrons in the electrode solution interface depends on the applied potential.

In a typical CV experiment, the potential is swept linearly with time from some starting potential, E_1, where species **Q** is stable to some other potential, E_2, at which electron transfer between species **Q** and the electrode is rapid, and species $\mathbf{Q}^+$ is formed. The potential is then swept back to E_1, causing electron transfer in the opposite direction and the reformation of **Q**. The rate of change of the potential from the initial potential, E_1 to the so called vertex potential, E_2, and back again is called the scan rate(ν in Vs^{-1}) [66]. This potential waveform is shown in Fig 3.1.

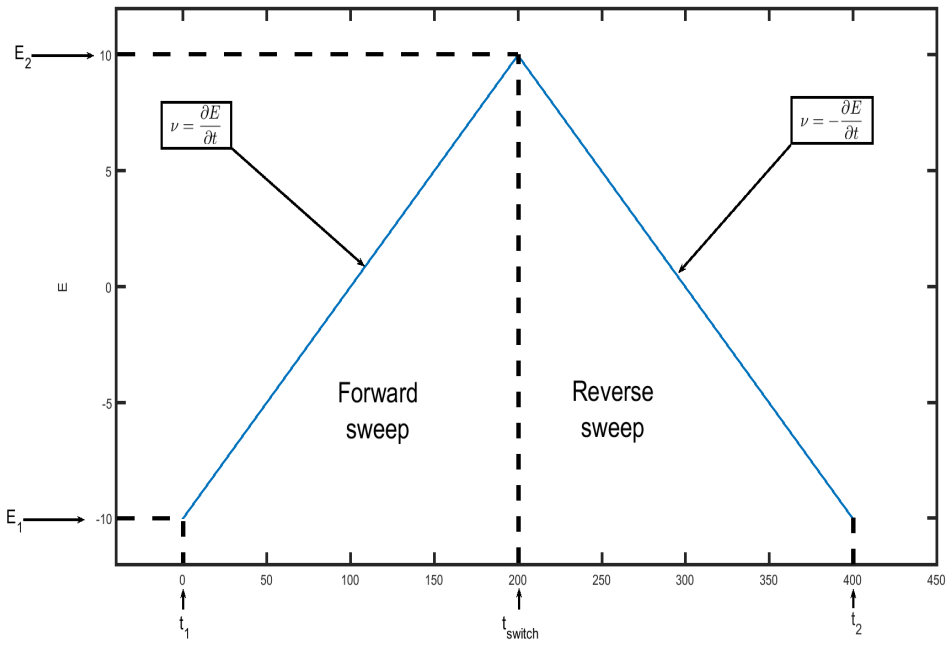


Figure 3.1: The waveform of the potential applied during a typical cyclic voltammetry experiment. In this case the initial potential, $E_1 = -10V$, and the vertex potential, $E_2 = 10V$, and the scan rate, $\nu = 0.1V.s^{-1}$.

On the forward sweep, the potential E, is given at any time t by $E = E_1 + \nu t$. At the time $t = t_{switch}$, the potential reach the reverse potential E_2 and change the direction. On the reverse sweep, the potential E, is given at any time $t > t_{switch}$ by $E = E_2 - \nu(t - t_{switch})$ or equivalently $E = 2E_2 - E_1 - \nu t$, since the time the potential change the direction t_{switch} is $(E_2 - E_1)/\nu$. The process can then be repeated in a periodic, or cyclic manner. Therefore, according to (3.2), the rate constants k_b and k_f are function of the time t (i.e. $k_b := k_b(t)$ and $k_f := k_f(t)$).

Throughout this process the current, I, (proportional to the rate of electron transfer) is recorded. We plot in Fig 3.2, the current-potential curve (or voltammogram) where I_{pc} and I_{pa} are called the peak cathodic and peak anodic current. The peak currents I_{pc} and I_{pa} are respectively associated the peak potential E_{pc} and E_{pa}. Note that in Fig 3.2, as the potential is scanned in the positive direction, the current rises to a peak and then decays. The current depends on two steps in the overall process, the movement of electroactive material to the surface and the electron transfer reaction.

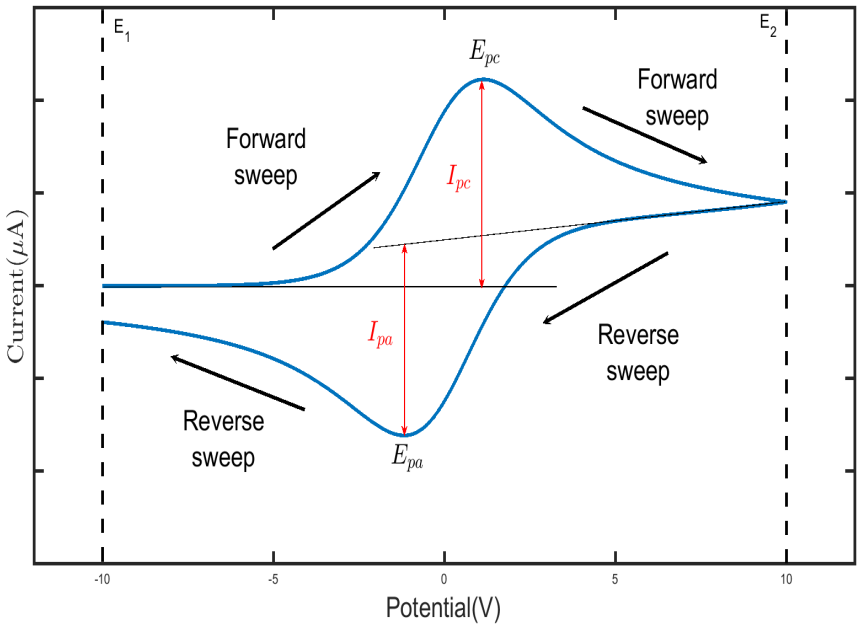


Figure 3.2: Voltammogram produced by the application of the potential waveform. We plot in this figure, a typical cyclic voltammogram where I_{pc} and I_{pa} show the peak cathodic and peak anodic current respectively associated the peak potential E_{pc} and E_{pa}

This technique is extremely useful experimentally as the resulting peak shaped

signal provides a direct fingerprint of the features of the reduction and oxidation processes. Analysis of the position and shape of the peaks can give important information about the nature of the electrochemical process taking place and about the chemical species themselves. The modelling of the CV experiment requires the definition of the electrical perturbation applied as well as the system under study, in term of mass transport, boundary conditions and heterogeneous or homogeneous chemical reaction. Therefore the mathematical problems faced in CV involve the resolution of a system of PDEs by means of analytical, semi-analytical or numerical methods. The solution of the problems are the concentration profiles of the species present in the chemical reaction, from which the voltammogram is deducted. Nevertheless it is not always feasible to use the analytical methods due to the complexity of the problems. The numerical methods offer a very accurate approximation to the true solution.

Unfortunately, simulation is usually obscure for non-theoreticians who often have to rely on software packages such as `pdepe` of MATLAB, which uses the FE method, described in [205], for the space discretization and ode15s algorithm, described in [201, 202], as time discretization. This will allow us to introduce the DG method in a way that allows any researcher or student to develop their own research and teaching tools for the study of voltammetry.

3.2 DG for electron transfer only model

We present here the numerical resolution (based on the DG discretization) of the electron transfer only (ETO) between the electrode and species that are chemically stable on the time scale of the experiment (3.1) with $n = 1$. Schematically, ETO mechanism is represented in Fig 3.3.

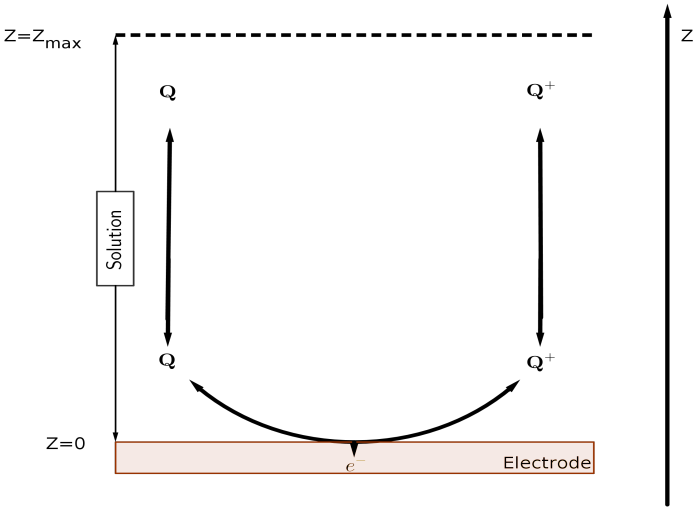


Figure 3.3: Electron transfer only mechanism.

3.2.1 Governing equations

The mathematical model to describe ETO is derived from the Fick's Law for mass transfer [69, 110, 109]. In one dimension, it is a coupled system of PDEs given by

$$\partial_t C_{\mathbf{Q}} = \partial_Z (D_{\mathbf{Q}} \partial_Z C_{\mathbf{Q}}), \qquad \forall (t, Z) \in [0, 2t_{switch}] \times [0, Z_{max}], \tag{3.3}$$

$$\partial_t C_{\mathbf{Q}^+} = \partial_Z (D_{\mathbf{Q}^+} \partial_Z C_{\mathbf{Q}^+}), \qquad \forall (t, Z) \in [0, 2t_{switch}] \times [0, Z_{max}], \tag{3.4}$$

subject to the boundary conditions

$$\partial_Z C_i(t, Z_{max}) = 0, \quad \forall t \in [0, 2t_{switch}], \ \forall i \in \{\mathbf{Q}, \mathbf{Q}^+\}, \tag{3.5}$$

$$D_{\mathbf{Q}} \partial_Z C_{\mathbf{Q}}(t, 0) = k_f(t) C_{\mathbf{Q}}(t, 0) - k_b(t) C_{\mathbf{Q}^+}(t, 0), \quad \forall t \in [0, 2t_{switch}], \tag{3.6}$$

$$D_{\mathbf{Q}^+} \partial_Z C_{\mathbf{Q}^+}(t, 0) = -k_f(t) C_{\mathbf{Q}}(t, 0) + k_b(t) C_{\mathbf{Q}^+}(t, 0), \quad \forall t \in [0, 2t_{switch}], \tag{3.7}$$

and the initial condition

$$C_i(0, Z) = C_i^0, \quad \forall Z \in [0, Z_{max}], \quad \forall i \in \{\mathbf{Q}, \mathbf{Q}^+\}, \tag{3.8}$$

where $C_i(t, Z) \in \mathbb{R}$ and $D_i \in \mathbb{R}$ are respectively the concentration and diffusion coefficients of the species $i \in \{\mathbf{Q}, \mathbf{Q}^+\}$. The current, I, is given by the flux of

electrons passing through the solution-electrode interface

$$I = -FA\partial_Z C_{\mathbf{Q}^+}\Big|_{Z=0}.$$

A useful way to analyse physical models in a more comprehensive manner is reducing the number of input variable affecting the model response in a way they can be easily reversed back to its physical properties. This can be done through dimensionless analysis, see [39] for more details.

Dimensionless parameters

For all species $i \in \{\mathbf{Q}, \mathbf{Q}^+\}$, the dimensionless concentration, $\tilde{C}_i$, is defined as the ratio between the concentration C_i and the sum C_m of the initial concentrations. Therefore, we have

$$\tilde{C}_i = \frac{C_i}{C_m},\ C_m = C^0_{\mathbf{Q}} + C^0_{\mathbf{Q}^+},\ \forall i \in \{\mathbf{Q}, \mathbf{Q}^+\}, \tag{3.9}$$

where C^0_i is the initial concentration of the specie $i \in \{\mathbf{Q}, \mathbf{Q}^+\}$. For the simulations, we consider $C^0_{\mathbf{Q}^+} = 0$, then we have $\tilde{C}^0_{\mathbf{Q}^+} = 0$ and $\tilde{C}^0_{\mathbf{Q}} = 1$. The dimensionless time $\tilde{t}$ is given by

$$\tilde{t} = \frac{t}{\tau}\,, \qquad \tau = \frac{RT}{F\nu}, \tag{3.10}$$

where the dimensionless parameter τ, used to make time dimensionless, represents the time needed to increase or decrease the potential by one thermal voltage.

Similarly to the time coordinate, the space coordinates can be made dimensionless by using, as the reference parameter, the diffusion length δ i.e. the distance in which the electro-active species moves within the reference time scale τ. Then, the dimensionless coordinates, z, is given by

$$z = \frac{Z}{\delta}, \qquad \delta = \sqrt{D_{\mathbf{Q}}\tau}. \tag{3.11}$$

The diffusion coefficients $D_{\mathbf{Q}}$ and $D_{\mathbf{Q}^+}$ are normalized by the diffusion coefficient of the electro-active species $\mathbf{Q}$ to calculate a dimensionless diffusion coefficient for

each species involve in chemical reaction

$$\tilde{D}_i = \frac{D_i}{D_{\mathbf{Q}}},\ \forall i \in \{\mathbf{Q}, \mathbf{Q}^+\}. \tag{3.12}$$

Then, the dimensionless diffusion coefficient $\tilde{D}_{\mathbf{Q}}$ is always equal one ($\tilde{D}_{\mathbf{Q}} = 1$). The dimensionless electron transfer rate constant given by

$$K_0 = k_0\sqrt{\frac{\tau}{D_{\mathbf{Q}}}}, \tag{3.13}$$

represents the level of reversibility of the system. The change in the applied potential, E, is converted to the dimensionless overpotential, P, by dividing the time coordinate by time scale τ. So, we have

$$P = \frac{F}{RT}(E - E^0). \tag{3.14}$$

Therefore, the dimensionless time step and dimensionless potential step coincide during the simulations of the dimensionless system. For the numerical simulation, we will assume that $E^0 = 0$.

Finally the current can be rearranged and its dimensionless form is given by

$$G = \frac{I}{AC_M F\sqrt{\frac{F\nu D_{\mathbf{Q}}}{RT}}} = \partial_z \tilde{C}_{\mathbf{Q}}\Big|_{z=0}. \tag{3.15}$$

Property 3.1. For the transfer coefficient $\alpha = 0.5$, it has been shown by Aoki et al. in [9] that the dimensionless peak cathodic current can be estimated by

$$G_{pc} = 0.446 + (0.247\sqrt{\alpha} - 0.223)(1 - \tanh(0.63\log(K_0) + \frac{0.189}{1-\alpha} - 0.219)), \tag{3.16}$$

with 1.2% relative errors. This shows that the reversibility of electron transfer depends on the parameters, namely K_0 and α, and is independent of $\tilde{D}_{Fc^+}$. $\Diamond$

Dimensionless governing equations

Substituting the dimensionless parameters into (3.3) to (3.8), we get the dimensionless governing equations of ETO given by

$$\partial_{\tilde{t}}\tilde{C}_{\mathbf{Q}} = \partial_z(\tilde{D}_{\mathbf{Q}}\partial_z\tilde{C}_{\mathbf{Q}}), \qquad \forall(\tilde{t},z)\in[0,2\tilde{t}_\lambda]\times[0,z_{max}], \tag{3.17}$$

$$\partial_{\tilde{t}}\tilde{C}_{\mathbf{Q}^+} = \partial_z(\tilde{D}_{\mathbf{Q}^+}\partial_z\tilde{C}_{\mathbf{Q}^+}), \qquad \forall(\tilde{t},z)\in[0,2\tilde{t}_\lambda]\times[0,z_{max}], \tag{3.18}$$

subject to the boundary conditions

$$\partial_z\tilde{C}_i(\tilde{t},z_{max}) = 0, \quad \forall\tilde{t}\in[0,2\tilde{t}_\lambda], \quad \forall i\in\{\mathbf{Q},\mathbf{Q}^+\} \tag{3.19}$$

$$\tilde{D}_{\mathbf{Q}}\partial_z\tilde{C}_{\mathbf{Q}}(\tilde{t},0) = K_f(\tilde{t})\tilde{C}_{\mathbf{Q}}(\tilde{t},0) - K_b(\tilde{t})\tilde{C}_{\mathbf{Q}^+}(\tilde{t},0), \quad \forall\tilde{t}\in[0,2\tilde{t}_\lambda], \tag{3.20}$$

$$\tilde{D}_{\mathbf{Q}^+}\partial_z\tilde{C}_{\mathbf{Q}^+}(\tilde{t},0) = -K_f(\tilde{t})\tilde{C}_{\mathbf{Q}}(\tilde{t},0) + K_b(\tilde{t})\tilde{C}_{\mathbf{Q}^+}(\tilde{t},0), \quad \forall\tilde{t}\in[0,2\tilde{t}_\lambda], \tag{3.21}$$

and the initial condition

$$\tilde{C}_i(0,z) = \tilde{C}_i^0, \quad \forall z\in[0,z_{max}], \quad \forall i\in\{\mathbf{Q},\mathbf{Q}^+\}. \tag{3.22}$$

The heterogeneous electron transfer rate constants can then be written in its dimensionless form as follows

$$K_f(\tilde{t}) = K_0\exp[(1-\alpha)P(\tilde{t})], \qquad K_b(\tilde{t}) = K_0\exp[(-\alpha)P(\tilde{t})], \tag{3.23}$$

where the dimensionless potential, P, in term of the dimensionless time, is given by

$$P(\tilde{t}) = \begin{cases} P_1+\tilde{t}, \ 0\le\tilde{t}\le\tilde{t}_\lambda \\ P_2-(\tilde{t}-\tilde{t}_\lambda), \ \tilde{t}_\lambda\le\tilde{t}\le 2\tilde{t}_\lambda \end{cases}, \qquad \tilde{t}_\lambda = P_2 - P_1, \tag{3.24}$$

with P_1 and P_2 respectively the dimensionless initial and reverse potential.

3.2.2 DG discretisation of the ETO model

Let $\cup_{i=1}^n I_i$, be a partition of the interval $\Omega = [0, z_{max}]$ into n elements with $I_i = [z_{i-1}, z_i]$. We associate to each element I_i, the step size h_i given by $h_i = z_i - z_{i-1}$

for all i in $\{1, \cdots, n\}$. This is illustrated in Fig 3.4.

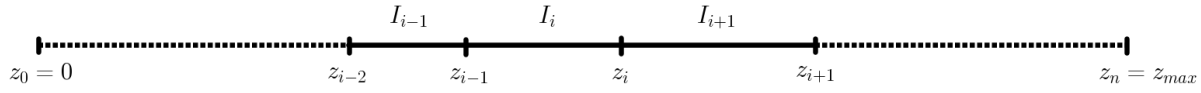


Figure 3.4: **Sketch of the one dimension domain Ω.**

The basic idea of the DG method is to approximate the solution locally on each element, $I_i,\ i \in \{1, \cdots, n\}$. Therefore integration by parts is applied locally and the test space for the variational formulation is defined as follows

$$V_h = \{u \in L^2(\Omega)\ :\ u\Big|_{I_i} \in \mathcal{P}^{k_i}(I_i), \forall\, i = \{1, \cdots, n\}\},$$

where $\mathcal{P}^{k_i}(I_i)$ is the set of polynomials of degree k_i on the interval I_i. Unless stated, we use $k_i = 2,\ \forall i \in \{1, \cdots, n\}$ for the numerical experiments.

Jump and average functions in 1D

For a given function $u \in V_h$, let us introduce the notations as follow

$$u(z) = u(\tilde{t}, z), \quad u(z_i^+) = \lim_{\substack{\epsilon \to 0 \\ \epsilon > 0}} u(z_i + \epsilon), \quad u(z_i^-) = \lim_{\substack{\epsilon \to 0 \\ \epsilon > 0}} u(z_i - \epsilon).$$

Then at an interior node z_i (i.e. $z_i,\ \ i = \{1, \cdots, n-1\}$), the jump function, $[u(z_i)]$, and the average function, $\{u(z_i)\}$, of u are given by

$$[u(z_i)] = u(z_i^-) - u(z_i^+), \qquad \{u(z_i)\} = \frac{1}{2}(u(z_i^-) + u(z_i^+)).$$

By convention, the definition of jump and average is extended to the external nodes. At the external nodes z_0 and z_n, the jump function is given by

$$[u(z_0)] = -u(z_0^+), \qquad [u(z_n)] = u(z_n^-),$$

and the average function is given by

$$\{u(z_0)\} = u(z_0^+), \qquad \{u(z_n)\} = u(z_n^-).$$

Formation of the weak problem of ETO

For simplicity, we introduce the following notation

$$D = \tilde{D}_{\mathbf{Q}}, \;\; D^+ = \tilde{D}_{\mathbf{Q}^+}, \;\; C = \tilde{C}_{\mathbf{Q}}, \;\; C^+ = \tilde{C}_{\mathbf{Q}^+}.$$

In order to derive the weak form of ETO, we will apply the methodology described in [80] to (3.17) and (3.18). Let us multiply (3.17) by u and then integrate by parts on each interval I_i to get

$$\int_{z_{i-1}}^{z_i} u\partial_{\tilde{t}} C dz + \int_{z_{i-1}}^{z_i} D\partial_z C \partial_z u dz - D\partial_z C(z_i^-)u(z_i^-) + D\partial_z C(z_{i-1}^+)u(z_{i-1}^+) = 0, \quad (3.25)$$

for all i in $\{1, \cdots, n\}$. By summing (3.25) over all values of i, we obtain

$$\sum_{i=1}^{n} \int_{z_{i-1}}^{z_i} u\partial_{\tilde{t}} C dz + \sum_{i=1}^{n} \int_{z_{i-1}}^{z_i} D\partial_z C \partial_z u dz - \sum_{i=1}^{n-1} [D\partial_z C(z_i)u(z_i)] + D\partial_z C(z_0^+)u(z_0^+) - D\partial_z C(z_n^-)u(z_n^-) = 0.$$

When we combine the above equation with the boundary conditions (3.19) and (3.20), we obtain

$$\sum_{i=1}^{n} \int_{z_{i-1}}^{z_i} u\partial_{\tilde{t}} C dz + \sum_{i=1}^{n} \int_{z_{i-1}}^{z_i} D\partial_z C \partial_z u dz - \sum_{i=1}^{n-1} [D\partial_z C(z_i)u(z_i)] + (K_f C(z_0) - K_b C^+(z_0))u(z_0) = 0. \quad (3.26)$$

For all $X_j, Y_j, j \in \{1, 2\}$, we have

$$X_1 Y_1 - X_2 Y_2 = \frac{1}{2}(Y_1 + Y_2)(X_1 - X_2) + \frac{1}{2}(X_1 + X_2)(Y_1 - Y_2). \quad (3.27)$$

Then letting $X_1 = D\partial_z C(z_i^-), X_2 = D\partial_z C(z_i^+), Y_1 = u(z_i^-)$ and $Y_2 = u(z_i^+)$ for all interior nodes $z_i, i \in \{1, \cdots, n-1\}$, yields

$$[D\partial_z C(z_i)u(z_i)] = \{D\partial_z C(z_i)\}[u(z_i)] + \{u(z_i)\}[D\partial_z C(z_i)]. \tag{3.28}$$

Substituting the equality (3.28) into (3.26) and noting that the exact solution C satisfies $[D\partial_z C(z_i)] = 0$ for all nodes z_i, $i \in \{1, \cdots, n-1\}$, we obtain

$$\sum_{i=1}^{n} \int_{z_{i-1}}^{z_i} u\partial_{\tilde{t}} C dz + \sum_{i=1}^{n} \int_{z_{i-1}}^{z_i} D\partial_z C \partial_z u dz - \sum_{i=1}^{n-1} \{D\partial_z C(z_i)\}[u(z_i)] \\ + (K_f(\tilde{t})C(z_0) - K_b(\tilde{t})C^+(z_0))u(z_0) = 0.$$

We now note that the exact solution also satisfies $[C(z_i)] = 0$ for all interior nodes z_i, i in $\{1, \cdots, n-1\}$. Therefore C and C^+ satisfy

$$\sum_{i=1}^{n} \int_{z_{i-1}}^{z_i} u\partial_{\tilde{t}} C dz + \sum_{i=1}^{n} \int_{z_{i-1}}^{z_i} D\partial_z C \partial_z u dz + (K_F C(z_0) - K_b C^+(z_0))u(z_0) \\ + \sum_{i=1}^{n-1} -\{D\partial_z C(z_i)\}[u(z_i)] + s\{D\partial_z u(z_i)\}[C(z_i)] + D\sigma_i[C(z_i)][u(z_i)] = 0 \tag{3.29}$$

where $\sigma_i = \sigma_0/h_{i,i+1}$, $\quad h_{i,i+1} = \max(h_i, h_{i+1})$, and the coefficients $\sigma_0, s \in \mathbb{R}$.

By applying the same methodology to (3.18), C and C^+ also satisfy

$$\sum_{i=1}^{n} \int_{z_{i-1}}^{z_i} u\partial_{\tilde{t}} C^+ dz + \sum_{i=1}^{n} \int_{z_{i-1}}^{z_i} D^+\partial_z C^+ \partial_z u dz - (K_F C(z_0) - K_b C^+(z_0))u(z_0) \\ + \sum_{i=1}^{n-1} -\{D^+\partial_z C^+(z_i)\}[u(z_i)] + s\{D^+\partial_z u(z_i)\}[C^+(z_i)] + \sigma_i[C^+(z_i)][u(z_i)] = 0. \tag{3.30}$$

Let us now define the following bilinear functions

$$\mathcal{A}_0(v,u) = \sum_{i=1}^{n} \int_{I_i} uv \, dz, \qquad \mathcal{B}_D(v,u) = D \sum_{i=1}^{n} \int_{I_i} \partial_z v \partial_z u \, dz, \tag{3.31}$$

$$\mathcal{C}_D^{s,\sigma_0}(v,u) = D \sum_{i=1}^{n-1} \underbrace{-\{\partial_z v(z_i)\}[u(z_i)] + s\{\partial_z u(z_i)\}[v(z_i)] + \sigma_i [v(z_i)][u(z_i)]}_{\mathcal{C}_i^{s,\sigma_0}(v,u)}, \tag{3.32}$$

$$\mathcal{D}_{K_f}(v,u) = -K_f v(z_0)u(z_0), \qquad \mathcal{E}_{K_b}(v,u) = -K_b v(z_0)u(z_0). \tag{3.33}$$

Then the equation (3.29) for C and (3.30) for C^+, can be rewritten as follows

$$\mathcal{A}_0(\partial_{\tilde{t}} C, u) + \overbrace{\mathcal{B}_D(C,u) + \mathcal{C}_D^{s,\sigma_0}(C,u) - \mathcal{D}_{K_f}(C,u)}^{\mathcal{A}_{D,K_f}^{s,\sigma_0}(C,u)} + \mathcal{E}_{K_b}(C^+,u) = 0 \tag{3.34}$$

$$\mathcal{A}_0(\partial_{\tilde{t}} C^+, u) + \underbrace{\mathcal{B}_{D^+}(C^+,u) + \mathcal{C}_{D^+}^{s,\sigma_0}(C^+,u) - \mathcal{E}_{K_b}(C^+,u)}_{\mathcal{A}_{D^+,K_b}^{s,\sigma_0}(C^+,u)} + \mathcal{D}_{K_f}(C,u) = 0 \tag{3.35}$$

Depending on the choices of the parameters σ_0 and s, we obtain different variations of DG methods that have appeared in the literature:

- If $s = -1$ and σ_0 is bounded below by a large enough constant, the resulting method, is called the symmetric interior penalty Galerkin (SIPG) method, introduced in 1978 by Wheeler [220] and Arnold [11]. It has also been shown that in one space dimension, the SIPG method for second order elliptic problems is stable for polynomial orders $k_i \geq 2$ using any stabilization parameter ($\sigma_0 = 0$) [42].

- If $s = 1$ and $\sigma_0 = 1$, the resulting method is called the non symmetric interior penalty Galerkin (NIPG) method, introduced in 1999 by Riviere, Wheeler and Girault [196].

In order to not increase the number of parameters of our system, we use NIPG for the simulations. We now examine in detail the derivation of the linear system of ODEs from the weak forms (3.34)-(3.35) and its implementation.

Linear system

To derive the linear system of ODEs from the weak forms (3.34)-(3.35), we first define the local basis function of $\mathcal{P}^{k_j}(I_j)$ from the shifted Legendre polynomials of degree less or equal to k_j on I_j. Since the application $z \longmapsto X(z) = 2(z-z_{j-1})/h_j - 1$ is a bijection from I_j onto $I = [-1, 1]$, so the shifted Legendre polynomials, ϕ_r^j, of degree r on I_j is defined as follow

$$\phi_r^j(z) = P_r\left(X(z)\right), \quad \forall z \in I_j, \tag{3.36}$$

where P_r is the Legendre polynomial of degree r on I. It is straightforward to see

$$\int_{I_j} \phi_r^j(z)\phi_l^j(z)dz = \frac{h_j}{2}\int_I P_r(x)P_l(x)dx \quad \text{and} \quad \| \phi_r^j \|_{L^2(I_j)} = \frac{\sqrt{h_j}}{\sqrt{2r+1}}. \tag{3.37}$$

For a given value of k_j, the set $\mathcal{P}^{k_j}(I_j)$ is then given by

$$\mathcal{P}^{k_j}(I_j) = \text{ span}\left\{ \frac{\phi_r^j}{\| \phi_r^j \|_{L^2(I_j)}},\ r \in \{0, \cdots, k_j\} \right\}.$$

For efficiency, we do not compute all the basis functions; but instead we only use the Legendre polynomials on the reference interval I. The orthonormal global basis functions of the finite space V_h, given by (3.38), are obtained from the orthonormal local basis functions by extending them with zero.

$$V_h = \text{span}\left\{\Phi_r^j, j \in \{1, \cdots, n\}, r \in \{0, \cdots, k_j\}\right\}, \quad \Phi_r^j(z) = \begin{cases} \frac{\phi_r^j(z)}{\|\phi_r^j\|_{L^2(I_j)}}, & z \in I_j \\ 0, & z \notin I_j \end{cases}. \tag{3.38}$$

For all z in Ω, we can expand the DG solutions C and C^+ and their time derivative as follow

$$C(z,\tilde{t}) = \sum_{j=1}^{n}\sum_{r=0}^{k_j} \alpha_r^j(\tilde{t})\Phi_r^j(z), \qquad \partial_{\tilde{t}} C(z,\tilde{t}) = \sum_{j=1}^{n}\sum_{r=0}^{k_j} d_{\tilde{t}}\alpha_r^j(\tilde{t})\Phi_r^j(z), \tag{3.39}$$

$$C^+(z,\tilde{t}) = \sum_{j=1}^{n}\sum_{r=0}^{k_j} \alpha_r^{+,j}(\tilde{t})\Phi_r^j(z), \qquad \partial_{\tilde{t}} C^+(z,\tilde{t}) = \sum_{j=1}^{n}\sum_{r=0}^{k_j} d_{\tilde{t}}\alpha_r^{+,j}(\tilde{t})\Phi_r^j(z), \tag{3.40}$$

where the time dependent coefficients α_r^j and $\alpha_r^{+,j}$ are the unknown variables to be solved for. Plugging (3.39), (3.40) into the variational formulations (3.34) and (3.35), we obtain for all $i \in \{1, \cdots, n\},\ l \in \{0, \cdots, k_i\}$

$$\sum_{j=1}^{n}\sum_{r=0}^{k_j} d_{\tilde{t}}\alpha_r^j \mathcal{A}_0(\Phi_r^j, \Phi_l^i) + \sum_{j=1}^{n}\sum_{r=0}^{k_j} \alpha_r^j \mathcal{A}_{D,K_f}^{s,\sigma_0}(\Phi_r^j, \Phi_l^i) + \sum_{j=1}^{n}\sum_{r=0}^{k_j} \alpha_r^{+,j} \mathcal{E}_{K_b}(\Phi_r^j, \Phi_l^i) = 0,$$
$$\sum_{j=1}^{n}\sum_{r=0}^{k_j} d_{\tilde{t}}\alpha_r^{+,j} \mathcal{A}_0(\Phi_r^j, \Phi_l^i) + \sum_{j=1}^{n}\sum_{r=0}^{k_j} \alpha_r^{+,j} \mathcal{A}_{D^+,K_b}^{s,\sigma_0}(\Phi_r^j, \Phi_l^i) + \sum_{j=1}^{n}\sum_{r=0}^{k_j} \alpha_r^{j} \mathcal{D}_{K_f}(\Phi_r^j, \Phi_l^i) = 0.$$

This is a linear system that can be rewritten as

$$\underbrace{\begin{pmatrix} M & 0 \\ 0 & M \end{pmatrix}}_{\mathcal{M}} \underbrace{\begin{pmatrix} d_{\tilde{t}}\alpha \\ d_{\tilde{t}}\alpha^+ \end{pmatrix}}_{d_{\tilde{t}}\chi} + \underbrace{\begin{pmatrix} A_{D,K_f}^{s,\sigma_0} & E_{K_b} \\ D_{K_f} & A_{D^+,K_b}^{s,\sigma_0} \end{pmatrix}}_{\mathcal{L}_{D,D^+,K_f,K_b}^{s,\sigma_0}} \underbrace{\begin{pmatrix} \alpha \\ \alpha^+ \end{pmatrix}}_{\chi} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{3.41}$$

where the vectors $\alpha,\ \alpha^+,\ d_t\alpha$ and $d_t\alpha^+$ are given by

$$\alpha = (\alpha_r^j), \quad \alpha^+ = (\alpha_r^{+,j}), \quad d_{\tilde{t}}\alpha = (d_{\tilde{t}}\alpha_r^j), \quad d_{\tilde{t}}\alpha^+ = (d_{\tilde{t}}\alpha_r^{+,j}),$$

and the block matrices $M,\ A_{D,K_f}^{s,\sigma_0},\ A_{D^+,K_b}^{s,\sigma_0},\ E_{K_b},\ D_{K_f}$ are given by

$$M = (\mathcal{A}_0(\Phi_r^j, \Phi_l^i)), \quad A_{D,K_f}^{s,\sigma_0} = (\mathcal{A}_{D,K_f}^{s,\sigma_0}(\Phi_r^j, \Phi_l^i)), \quad A_{D^+,K_b}^{s,\sigma_0} = (\mathcal{A}_{D^+,K_b}^{s,\sigma_0}(\Phi_r^j, \Phi_l^i)),$$
$$E_{K_b} = (\mathcal{E}_{K_b}(\Phi_r^j, \Phi_l^i)), \quad D_{K_f} = (\mathcal{D}_{K_f}(\Phi_r^j, \Phi_l^i)),$$

where $i, j \in \{1, \cdots, n\}$, $r \in \{0, \cdots, k_j\}$ and $l \in \{0, \cdots, k_i\}$.

The matrices $\mathcal{M}$ and $\mathcal{L}_{D,D^+,K_f,K_b}^{s,\sigma_0}$ are respectively called the mass matrix and the stiffness matrix and due to the local definition of the basis functions, most entries of their block matrices are equal to zeros.

Assembly of the mass matrix

In order to assemble the mass matrix, $\mathcal{M}$, we only need to assemble the block matrix $M = (M^{I_jI_i})$, where $M^{I_jI_i}$ is a $(k_i+1) \times (k_j+1)$ matrix, for all $i, j \in \{1, \cdots, n\}$, with entries $\mathcal{A}_0(\Phi_r^j, \Phi_l^i)$, $r \in \{0, \cdots, k_j\}$ and $l \in \{0, \cdots, k_i\}$. Since the basis function

$\Phi_r^j(z) = 0$ for all $z \notin I_j$ and are orthonormal, then from (3.31) we obtain

$$\mathcal{A}_0(\Phi_r^j, \Phi_l^i) = \begin{cases} 0 & \text{if } i \neq j,\ r \in \{0, \cdots, k_j\},\ l \in \{0, \cdots, k_i\} \\ \int_{I_j} \Phi_r^j \Phi_l^j = \delta_r^l & \text{if } i = j,\ r \in \{0, \cdots, k_j\},\ l \in \{0, \cdots, k_i\} \end{cases}. \quad (3.42)$$

Therefore the block matrix M is a $n_T \times n_T$ identity matrix and $\mathcal{M}$ is $(2n_T) \times (2n_T)$ identity matrix, where n_T is the dimension of the finite space V_h, given by

$$n_T = \sum_{j=1}^{n} (k_j + 1). \quad (3.43)$$

Assembly of stiffness matrix

In order to assemble the stiffness matrix, $\mathcal{L}^{s,\sigma_0}_{D,D^+,K_f,K_b}$, we need to assemble the block matrices $E_{K_b}, D_{K_f}, A^{s,\sigma_0}_{D,K_f}$ and $A^{s,\sigma_0}_{D^+,K_b}$. First we investigate the assembly of E_{K_b} and D_{K_f}. Since the entries of the first two block matrices depend only on the boundary $z_0 = 0$, then their entries will be different to zero only for the basis functions defined locally on the element I_1. We have for $r, l \in \{0, \cdots, k_1\}$

$$\mathcal{D}_{K_f}(\Phi_r^1, \Phi_l^1) = -K_f \Phi_r^1(0)\Phi_l^1(0), \qquad \mathcal{E}_{K_b}(\Phi_r^1, \Phi_l^1) = -K_b \Phi_r^1(0)\Phi_l^1(0). \quad (3.44)$$

According to the definition of the global basis functions (see (3.36) and (3.38)), the above bilinear functions are equivalent to

$$\mathcal{D}_{K_f}(\Phi_r^j, \Phi_l^i) = -K_f P_r(-1)P_l(-1)\Gamma^1_{\{r,l\}}, \quad \mathcal{E}_{K_b}(\Phi_r^1, \Phi_l^1) = -K_b P_r(-1)P_l(-1)\Gamma^1_{\{r,l\}},$$

where Γ_S^j is given by

$$\Gamma_S^j = \prod_{r \in S} \frac{1}{\| \phi_r^j \|_{L^2(I_j)}} = \prod_{r \in S} \frac{\sqrt{2r+1}}{\sqrt{h_j}} = \sqrt{h_j^{-N_S}} \underbrace{\prod_{r \in S} \sqrt{2r+1}}_{\Gamma_S}. \quad (3.45)$$

for a given j in $\{1, \cdots, n\}$ and set S with N_S elements. More generally, the non zero entries of the block matrix E_{K_b} and D_{K_f} are given by

$$\mathcal{E}_{K_b}(\Phi_r^1, \Phi_l^1) = -(-1)^{r+l}\Gamma^1_{\{r,l\}}K_b(\tilde{t}), \quad \mathcal{D}_{K_b}(\Phi_r^1, \Phi_l^1) = -(-1)^{r+l}\Gamma^1_{\{r,l\}}K_f(\tilde{t})$$

Therefore the matrices E_{K_b} and D_{K_f} are defined as follow

$$E_{K_b} = -K_b(\tilde{t}) \underbrace{\begin{pmatrix} F_1 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix}}_{L1}, \quad D_{K_f} = -K_f(\tilde{t}) \underbrace{\begin{pmatrix} F_1 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix}}_{L1}, \tag{3.46}$$

where F_1 is $(k_1+1)\times(k_1+1)$ squared matrix with entry $F_1(r,l) = (-1)^{r+l}\Gamma^1_{\{r,l\}}$.

Finally we investigate the assembly of A^{s,σ_0}_{D,K_f} and $A^{s,\sigma_0}_{D^+,K_b}$. Let us introduce two bilinear function $\mathcal{B}$ and $\mathcal{C}^{s,\sigma_0}$ such that for any function u, v in V_h

$$\mathcal{B}_D(u,v) = D\mathcal{B}(u,v), \qquad \mathcal{C}^{s,\sigma_0}_D(u,v) = D\mathcal{C}^{s,\sigma_0}(u,v). \tag{3.47}$$

Then the entries of the the matrices A^{s,σ_0}_{D,K_f} and $A^{s,\sigma_0}_{D^+,K_b}$ are given by

$$\mathcal{A}^{s,\sigma_0}_{D,K_f}(\Phi_r^j, \Phi_l^i) = D(\mathcal{B}(\Phi_r^j, \Phi_l^i) + \mathcal{C}^{s,\sigma_0}(\Phi_r^j, \Phi_l^i)) - \mathcal{D}_{K_f}(\Phi_r^j, \Phi_l^i) \tag{3.48}$$

$$\mathcal{A}^{s,\sigma_0}_{D^+,K_f}(\Phi_r^j, \Phi_l^i) = D^+(\mathcal{B}(\Phi_r^j, \Phi_l^i) + \mathcal{C}^{s,\sigma_0}(\Phi_r^j, \Phi_l^i)) - \mathcal{E}_{K_b}(\Phi_r^j, \Phi_l^i), \tag{3.49}$$

for all $i, j \in \{1, \cdots, n\}$, $r \in \{0, \cdots, k_j\}$, $l \in \{0, \cdots, k_i\}$. Since the matrices E_{K_b} and D_{K_f} have previously been investigated, all that remains to do is to assemble the spatial contribution matrix, B, with entries $\mathcal{B}(\Phi_r^j, \Phi_l^i)$ and the interior nodal contribution matrix, C^{s,σ_0}, with entries $\mathcal{C}^{s,\sigma_0}(\Phi_r^j, \Phi_l^i)$. Then we can evaluate the matrices A^{s,σ_0}_{D,K_f} and $A^{s,\sigma_0}_{D^+,K_b}$ as follow

$$A^{s,\sigma_0}_{D,K_f} = D(B + C^{s,\sigma_0}) - D_{K_f}, \qquad A^{s,\sigma_0}_{D^+,K_b} = D^+(B + C^{s,\sigma_0}) - E_{K_b}.$$

1. Spatial contribution matrix B: Due to the extension of the local basis function to global, (3.38), we have

$$\mathcal{B}(\Phi_r^j, \Phi_l^i) = \int_{I_j \cap I_i} d_z\phi_r^j(z) d_z\phi_l^i(z)\, dz.$$

Therefore $\mathcal{B}(\Phi_r^j, \Phi_l^i)$ is equal to zero if $i \neq j$, since $I_i, i = 1, \cdots, n$ represent a partition. Using the set of reference basis functions, $\{P_r, r \geq 0\}$, on the interval $I = [-1, 1]$, we obtain

$$\mathcal{B}(\Phi_r^i, \Phi_l^i) = \Gamma^i_{\{r,l\}} \int_{I_i} d_z\phi_r^i(z) d_z\phi_l^i(z)\, dz = \frac{2}{h_i^2} \underbrace{\left(\Gamma_{\{r,l\}} \int_I d_x P_r(x) d_x P_l(x)\, dx \right)}_{\mathcal{B}_1(P_r, P_l)}.$$

The spatial contribution matrix then takes the form

$$B = \begin{pmatrix} \frac{2}{h_1^2} B^{I_1} & 0 & \cdots & 0 \\ 0 & \frac{2}{h_2^2} B^{I_2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \frac{2}{h_n^2} B^{I_n} \end{pmatrix},$$

where B^{I_i} is a $(k_i+1)\times(k_i+1)$ symmetric matrix with entries $\mathcal{B}_1(P_r, P_l)$ for all $r, l = 0, \cdots, k_i$. All block matrices B^{I_i} can be extracted from $B^{I_{max}}$ associated to $k_{max} = \max\{k_i, i = 1, \cdots, n\}$. To assemble $B^{I_{max}}$ efficiently, we only need to compute its entries for all $l = 0, \cdots, k_{max}$ and $l \leq r$. Using the Legendre polynomials properties (A.5) and (A.6), we have for $l \leq r,\ l = 0, \cdots, k_{max}$

$$B^{I_{max}}(r, l) = \begin{cases} 0 & \text{if } r + l \text{ is odd} \\ l(l+1)\Gamma_{\{r,l\}}, & \text{otherwise} \end{cases}. \tag{3.50}$$

2. Interior nodal contribution matrix C^{s,σ_0}: Due to the extension of the local basis function to global, (3.38), we have

$$\mathcal{C}^{s,\sigma_0}(\Phi_r^j, \Phi_l^i) \neq 0 \quad \text{if and only if} \quad I_i \cap I_j \neq \emptyset.$$

Then the interior nodal contribution matrix takes the form

$$C^{s,\sigma_0} = \begin{pmatrix} C^{s,\sigma_0}_{I_1I_1} & C^{s,\sigma_0}_{I_2I_1} & 0 & \cdots & 0 \\ C^{s,\sigma_0}_{I_1I_2} & C^{s,\sigma_0}_{I_2I_2} & C^{s,\sigma_0}_{I_3I_2} & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & C^{s,\sigma_0}_{I_{n-2}I_{n-1}} & C^{s,\sigma_0}_{I_{n-1}I_{n-1}} & C^{s,\sigma_0}_{I_nI_{n-1}} \\ 0 & \cdots & 0 & C^{s,\sigma_0}_{I_{n-1}I_n} & C^{s,\sigma_0}_{I_nI_n} \end{pmatrix},$$

where the block matrix $C^{s,\sigma_0}_{I_jI_i}$ is a $(k_i+1)\times(k_j+1)$ matrix, which according to (3.32), is defined as follows

$$C^{s,\sigma_0}_{I_jI_i} = \left(\mathcal{C}^{s,\sigma_0}(\Phi^j_r,\Phi^i_l)\right)^{l=0,\cdots,k_i}_{r=0,\cdots,k_j}, \quad \mathcal{C}^{s,\sigma_0}(\Phi^j_r,\Phi^i_l) = \sum_{m=1}^{n-1}\mathcal{C}^{s,\sigma_0}_m(\Phi^j_r,\Phi^i_l)$$

The bilinear function $\mathcal{C}^{s,\sigma_0}_i$ is defined at the interior node z_i and each interior node z_i contributes to four blocks of the global matrix, C^{s,σ_0}, which are

$$A^i = \begin{pmatrix} C^{s,\sigma_0}_{I_iI_i} & C^{s,\sigma_0}_{I_{i+1}I_i} \\ C^{s,\sigma_0}_{I_iI_{i+1}} & C^{s,\sigma_0}_{I_{i+1}I_{i+1}} \end{pmatrix}.$$

The local stiffness matrix stemming from the interior node can be block-partitioned in the form

$$C^i = \begin{pmatrix} C^i_{I_iI_i} & C^i_{I_{i+1}I_i} \\ C^i_{I_iI_{i+1}} & C^i_{I_{i+1}I_{i+1}} \end{pmatrix}, \quad C^i_{I_{r1}I_{r2}} = \left(\mathcal{C}^{s,\sigma_0}_i(\Phi^{r1}_r,\Phi^{r2}_l)\right)^{l=0,\cdots,k_{r2}}_{r=0,\cdots,k_{r1}}, \quad r1,r2\in\{i,i{+}1\}$$

such that the matrix A^i is updated by

$$A^i \longrightarrow A^i + C^i, \quad \text{for } \; i = 1,\cdots,n-1.$$

Combining the formula of the bilinear function $\mathcal{C}_i^{s,\sigma_0}$, in (3.32), the definition of our global basis function in (3.36), (3.38) and the properties of Legendre polynomials in (A.1), (A.6); we obtain

$$\begin{aligned}
\mathcal{C}_i^{s,\sigma_0}(\Phi_r^i, \Phi_l^i) &= \left(-\frac{r(r+1)}{2h_i} + s\frac{l(l+1)}{2h_i} + \sigma_i\right)\Gamma^i_{\{r,l\}} \\
\mathcal{C}_i^{s,\sigma_0}(\Phi_r^i, \Phi_l^{i+1}) &= \left(\frac{r(r+1)}{2h_i} - s\frac{l(l+1)}{2h_{i+1}} - \sigma_i\right)(-1)^l\Gamma^i_{\{r\}}\Gamma^{i+1}_{\{l\}} \\
\mathcal{C}_i^{s,\sigma_0}(\Phi_r^{i+1}, \Phi_l^i) &= \left(\frac{r(r+1)}{2h_{i+1}} - s\frac{l(l+1)}{2h_i} - \sigma_i\right)(-1)^r\Gamma^{i+1}_{\{r\}}\Gamma^{i}_{\{l\}} \\
\mathcal{C}_i^{s,\sigma_0}(\Phi_r^{i+1}, \Phi_l^{i+1}) &= \left(-\frac{r(r+1)}{2h_{i+1}} + s\frac{l(l+1)}{2h_{i+1}} + \sigma_i\right)(-1)^{l+r}\Gamma^{i+1}_{\{r,l\}}
\end{aligned} \tag{3.51}$$

For more details about evaluating these quantities, see Section A.3.

By taking $L_0^{s,\sigma_0} = B + C^{s,\sigma_0}$, the stiffness matrix is then defined as follow

$$\mathcal{L}^{s,\sigma_0}_{D,D^+,K_f,K_b} = \underbrace{\left(\begin{array}{c|c} DL_0^{s,\sigma_0} & 0 \\ \hline 0 & D^+L_0^{s,\sigma_0} \end{array}\right)}_{\mathcal{L}_0} + \overbrace{\left(\begin{array}{c|c} K_f(\tilde{t})L1 & -K_b(\tilde{t})L1 \\ \hline -K_f(\tilde{t})L1 & K_b(\tilde{t})L1 \end{array}\right)}^{\mathcal{L}_1(\tilde{t})}. \tag{3.52}$$

Note from (3.52) that the matrix $\mathcal{L}_0$, unlike $\mathcal{L}_1$, is independent of the dimensionless time $\tilde{t}$. Therefore only the matrix $\mathcal{L}_1$ need to be updated during the resolution of the ODEs (3.41) by the time integrators such as Impl or ETD.

3.2.3 Computation of the concentrations and the current

The system of PDEs (3.17), (3.18) subject to the boundary conditions from (3.19) to (3.21) that describe ETO model, is transformed into the system of ODEs (3.41), which can be rewritten as follows

$$\frac{d\chi}{d\tilde{t}} = -\mathcal{M}^{-1}(\mathcal{L}_0 + \mathcal{L}_1(\tilde{t}))\chi = -(\mathcal{L}_0 + \mathcal{L}_1(\tilde{t}))\chi, \tag{3.53}$$

since $\mathcal{M}$ is the identity matrix. The matrices $\mathcal{L}_0, \mathcal{L}_1$ are defined in (3.52). The initial condition (3.22) is equivalent to

$$\chi\Big|_{\tilde{t}=0} = \chi_0 = (\alpha_0, \alpha_0^+)^T, \tag{3.54}$$

where the entries of the vectors α_0 and α_0^+ are respectively given by

$$\alpha_{0,r}^i = \begin{cases} \tilde{C}_{\mathbf{Q}}^0 \sqrt{h_i} & \text{if } r = 0 \\ 0, & \text{otherwise} \end{cases}, \quad \alpha_{0,r}^{+,i} = \begin{cases} \tilde{C}_{\mathbf{Q}^+}^0 \sqrt{h_i} & \text{if } r = 0 \\ 0, & \text{otherwise} \end{cases}, \tag{3.55}$$

for all $i = 1, \cdots, n$ and $r = 0, \cdots, k_i$. For more details about evaluating these quantities, see Section A.2.

The time dependent ODEs of (3.53), can be identified with (2.10), where the matrix L and the function F are defined as follows

$$L \equiv L(\tilde{t}) = -(\mathcal{L}_0 + \mathcal{L}_1(\tilde{t})), \quad F = 0. \tag{3.56}$$

Since the matrix L depends on the time, $\tilde{t}$, the system of ODEs (3.53) subject to the initial condition (3.54) can be solved with the time discretization method discussed in Chapter 2, with the following approximation

$$\forall \tilde{t} \in [\tilde{t}_n, \tilde{t}_{n+1}], \qquad L \approx L(\tilde{t}_{n+1}).$$

Throughout the resolution of the system of ODEs (3.53), the dimensionless current is simultaneously computed with the formula

$$G(\tilde{t}) = K_f(\tilde{t}) \sum_{r=0}^{k_1} (-1)^r \alpha_r^1(\tilde{t}) \Gamma_{\{r\}}^1 - K_b(\tilde{t}) \sum_{r=0}^{k_1} (-1)^r \alpha_r^{+,1}(\tilde{t}) \Gamma_{\{r\}}^1, \tag{3.57}$$

where the $\chi(\tilde{t}) = (\alpha(\tilde{t}), \alpha^+(\tilde{t}))^T$ is the solution of system of ODEs (3.53) subject to the initial condition (3.54) at the time $\tilde{t}$.

3.2.4 Numerical experiments on ETO model

We realise some numerical experiments using MATLAB's solver (`pdepe`) and our solvers (DG method combined with Implicit Euler method or ETD_1), in order to validate our time and space discretization and find the most appropriate solver for the ETO model. Unless stated, the time solver ETD_1 is implemented with the use of the MATLAB's code *phipm* to compute $e^W X$, where W is a square matrix and X

a vector, see for example [172, 217] for more details. By default we also set $K_0 = 20$ and $\tilde{D}_{\mathbf{Q}^+} = 1$. Then according (3.16), the dimensionless peak cathodic current is $G_{pc}^{20} = 0.4444$.

Numerical simulation of the concentration profile

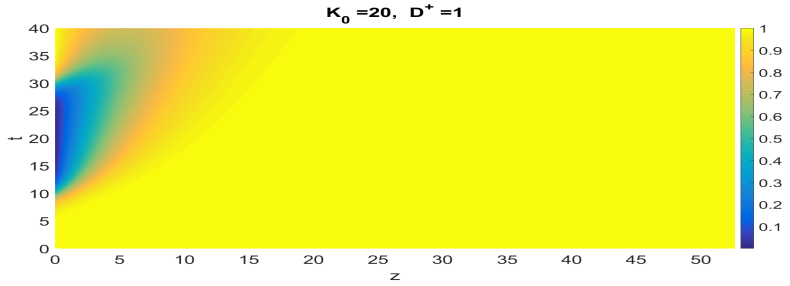
In order to validate our space discretization, we simulate the concentration profile of $\mathbf{Q}$ and $\mathbf{Q}^+$, using the DG method combined with the Implicit Euler method. If the species $\mathbf{Q}$ and $\mathbf{Q}^+$ have the same diffusion coefficient, then according to the dimensionless governing equations (3.17) to (3.22), the sum, $\tilde{S}$ of the dimensionless concentration of the species $\mathbf{Q}$ and $\mathbf{Q}^+$ is subject to the following PDEs

$$\partial_{\tilde{t}}\tilde{S} = \partial_z(\tilde{D}_{\mathbf{Q}}\partial_z\tilde{S}), \quad \partial_z\tilde{S}\Big|_{z\in\{0,z_{max}\}} = 0, \quad \tilde{S}\Big|_{\tilde{t}=0} = 1. \tag{3.58}$$

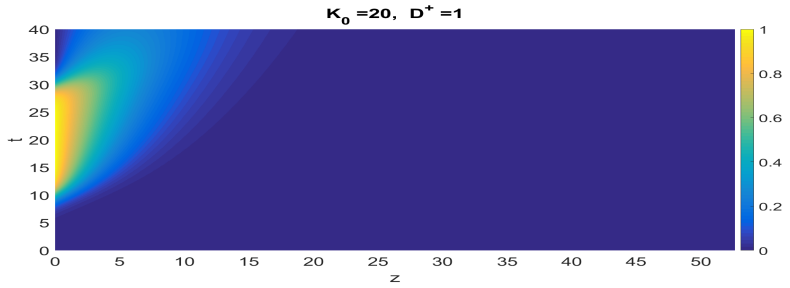
Since $\tilde{C}_m$ is a constant, we have at any time t and any point z

$$\tilde{S}(\tilde{t}, z) = \tilde{C}_{\mathbf{Q}}(\tilde{t}, z) + \tilde{C}_{\mathbf{Q}^+}(\tilde{t}, z) = 1. \tag{3.59}$$

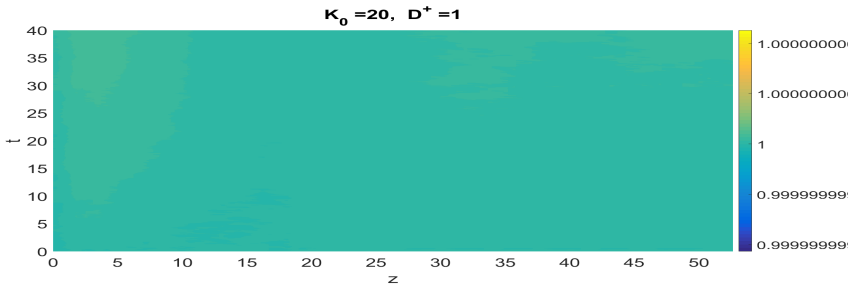
We use the implicit Euler method to solve (3.53) (obtained with the DG discretization) subject to (3.54) for $\tilde{D}_{\mathbf{Q}} = \tilde{D}_{\mathbf{Q}^+} = 1$ and $K_0 = 20$. We then show for all $\tilde{t}, z$ the dimensionless concentration $\tilde{C}_{\mathbf{Q}}$ in Fig 3.5(a), the dimensionless concentration $\tilde{C}_{\mathbf{Q}^+}$ in Fig 3.5(b) and the dimensionless concentration $\tilde{S}$ in Fig 3.5(c). Note that in Fig 3.5(c), we see that $\tilde{S} = 1, \forall x, \tilde{t}$ as expected.



(a) Dimensionless concentration profile $\tilde{C}_{\mathbf{Q}}$



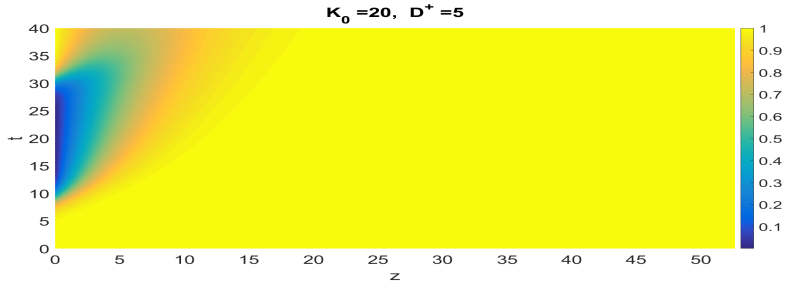
(b) Dimensionless concentration profile $\tilde{C}_{\mathbf{Q}^+}$



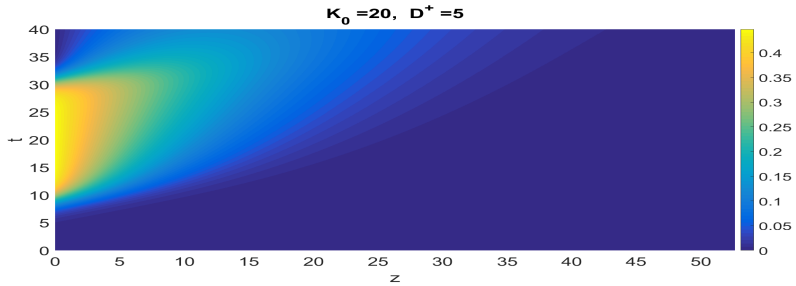
(c) Dimensionless concentration profile $\tilde{S}$

Figure 3.5: **Dimensionless concentration profile of $\mathbf{Q}, \mathbf{Q}^+$ and their sum $\tilde{S}$ for $\tilde{D}_{\mathbf{Q}} = \tilde{D}_{\mathbf{Q}^+} = 1$ and $K_0 = 20$.** We respectively show for all $\tilde{t}, z$ the dimensionless concentration $\tilde{C}_{\mathbf{Q}}, \tilde{C}_{\mathbf{Q}^+}$ and $\tilde{S}$ in (a), (b) and (c). Note that in (c), we see that $\tilde{S} = 1, \forall x, \tilde{t}$ as expected.

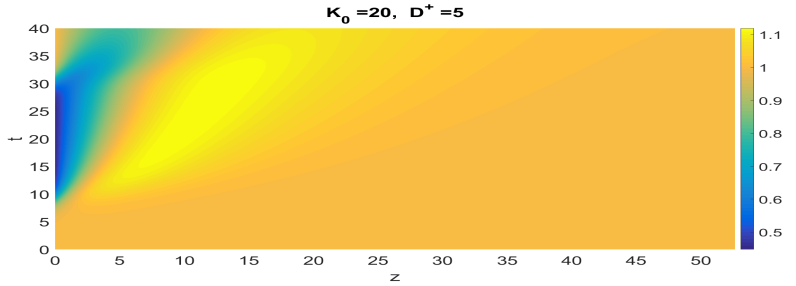
If the $\mathbf{Q}$ and $\mathbf{Q}^+$ do not have equal diffusion coefficient, then the dimensionless governing equations (3.17) to (3.22) do not hold any more. We use the implicit Euler method to solve (3.53) (obtained with the DG discretization) subject to (3.54) for $\tilde{D}_{\mathbf{Q}} = 1, \tilde{D}_{\mathbf{Q}^+} = 5$ and $K_0 = 20$. We then show for all $\tilde{t}, z$ the dimensionless concentration $\tilde{C}_{\mathbf{Q}}$ in Fig 3.6(a), the dimensionless concentration $\tilde{C}_{\mathbf{Q}^+}$ in Fig 3.6(b) and the dimensionless concentration $\tilde{S}$ in Fig 3.6(c). Note that in Fig 3.6(c), we see that $\exists x, \tilde{t},\ \tilde{S} \neq 1$ as expected.



(a) Dimensionless concentration profile $\tilde{C}_{\mathbf{Q}}$



(b) Dimensionless concentration profile $\tilde{C}_{\mathbf{Q}^+}$



(c) Dimensionless concentration profile $\tilde{S}$

Figure 3.6: **Dimensionless concentration profile of $\mathbf{Q}, \mathbf{Q}^+$ and their sum $\tilde{S}$ for $\tilde{D}_{\mathbf{Q}} = 1, \tilde{D}_{\mathbf{Q}^+} = 5$ and $K_0 = 20$.** We respectively show for all $\tilde{t}, z$ the dimensionless concentration $\tilde{C}_{\mathbf{Q}}, \tilde{C}_{\mathbf{Q}^+}$ and $\tilde{S}$ in (a), (b) and (c). Note that in (c), we see that $\exists x, \tilde{t},\ \tilde{S} \neq 1$ as expected.

Even if (3.17) to (3.22) do not hold, the conservation law still holds i.e.

$$\int_\Omega \tilde{C}_{\mathbf{Q}}(\tilde{t}, z) + \tilde{C}_{\mathbf{Q}^+}(\tilde{t}, z)dz = \int_\Omega \tilde{C}_{\mathbf{Q}}(0, z) + \tilde{C}_{\mathbf{Q}^+}(0, z)dz = z_{max}. \tag{3.60}$$

Then at any time $\tilde{t}$, we have

$$\tilde{\mathcal{R}}(\tilde{t}) = \frac{1}{z_{max}} \int_\Omega \tilde{C}_{\mathbf{Q}}(\tilde{t}, z) + \tilde{C}_{\mathbf{Q}^+}(\tilde{t}, z)dz = 1. \tag{3.61}$$

Throughout the computation of the dimensionless concentrations of $\mathbf{Q}$ and $\mathbf{Q}^+$, we also compute the $\tilde{\mathcal{R}}(\tilde{t})$. We then plot $\tilde{\mathcal{R}}$ as a function of $\tilde{t}$ in Fig 3.7. Note that

$\tilde{\mathcal{R}}(\tilde{t}) = 1, \forall \tilde{t}$ as expected.

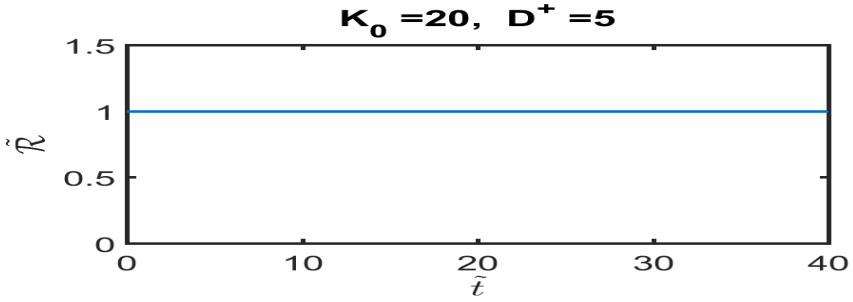


Figure 3.7: **Function $\tilde{\mathcal{R}}$.** Throughout the computation of the dimensionless concentrations of $\mathbf{Q}$ and $\mathbf{Q}^+$, we also compute the $\tilde{\mathcal{R}}(\tilde{t})$. This plot shows $\tilde{\mathcal{R}}$ as a function of $\tilde{t}$. Note that in this plot, $\tilde{\mathcal{R}}(\tilde{t}) = 1, \forall \tilde{t}$ as expected.

We compare the convergence and the efficiency of the solvers `pdepe`, DG method combined with implicit Euler method (DG_{Impl}) and DG method combined with ETD1 (DG_{ETD_1}), with respect to the time and space discretization, while computing the concentration profile of the species $\mathbf{Q}$ and $\mathbf{Q}^+$ at the final time $\tilde{t}_2 = 2(P_2 - P_1)$.

Firstly, to examine the convergence and the efficiency with respect to the time discretization, we compute the dimensionless concentration profile of the species $\mathbf{Q}$ and $\mathbf{Q}^+$ at the final time, using *pedpe* , DG_{Impl} and DG_{ETD_1}, for the each time step $\Delta t_i = 2^i \Delta t_0$, for $i = 0, \cdots, 5$. By considering the concentration profile associated to the finest time step Δt_0 as exact concentration, we compute the relative error associated to the time step Δt_i, $i \in \{1, \cdots, 5\}$ by

$$E_i^{1,r} = 100 \frac{\sqrt{\sum_{j=\mathbf{Q},\mathbf{Q}^+} \left| C_j^{0,r} - C_j^{i,r} \right|^2_{L^2(\Omega)}}}{\sqrt{\sum_{j=\mathbf{Q},\mathbf{Q}^+} \left| C_j^{0,r} \right|^2_{L^2(\Omega)}}}, \tag{3.62}$$

for the each solver $r = \texttt{pdepe}, DG_{Impl}, DG_{ETD_1}$. We respectively plot $\tilde{C}_{\mathbf{Q}}$ and $\tilde{C}_{\mathbf{Q}^+}$ at final time $\tilde{t}_2$ for $\Delta t_0 = 3.8147 \times 10^{-5}$ in Fig 3.8(a) and Fig 3.8(b). The convergence and the efficiency in this case are respectively illustrated by plotting $E_i^{1,r}$ vs Δt_i in Fig 3.8(c) and $E_i^{1,r}$ vs CPU time in Fig 3.8(d). Note that in Fig 3.8(c), the relative error $E_i^{1,r}$ decrease with the time step and the curves associated to the three solvers are parallel. We can conclude that the solvers *pedpe*, DG_{Impl}, DG_{ETD_1} converge in time and have the same order of convergence with respect to the time step. Note

that in Fig 3.8(d), for a given relative E^0, we have

$$T^0_{DG_{ETD_1}} > T^0_{\texttt{pdepe}} > T^0_{DG_{Impl}}, \tag{3.63}$$

where T^0_r is the CPU time spends by the solver $r = \texttt{pdepe}, DG_{Impl}, DG_{ETD_1}$ for the computation of $\tilde{C}_{\mathbf{Q}}$ and $\tilde{C}_{\mathbf{Q}^+}$ at the time $\tilde{t} = \tilde{t}_2$ such that $E^{1,r}_{i_0} = E^0$. Thus DG_{Impl} is the most efficient solver with respect to the time discretization.

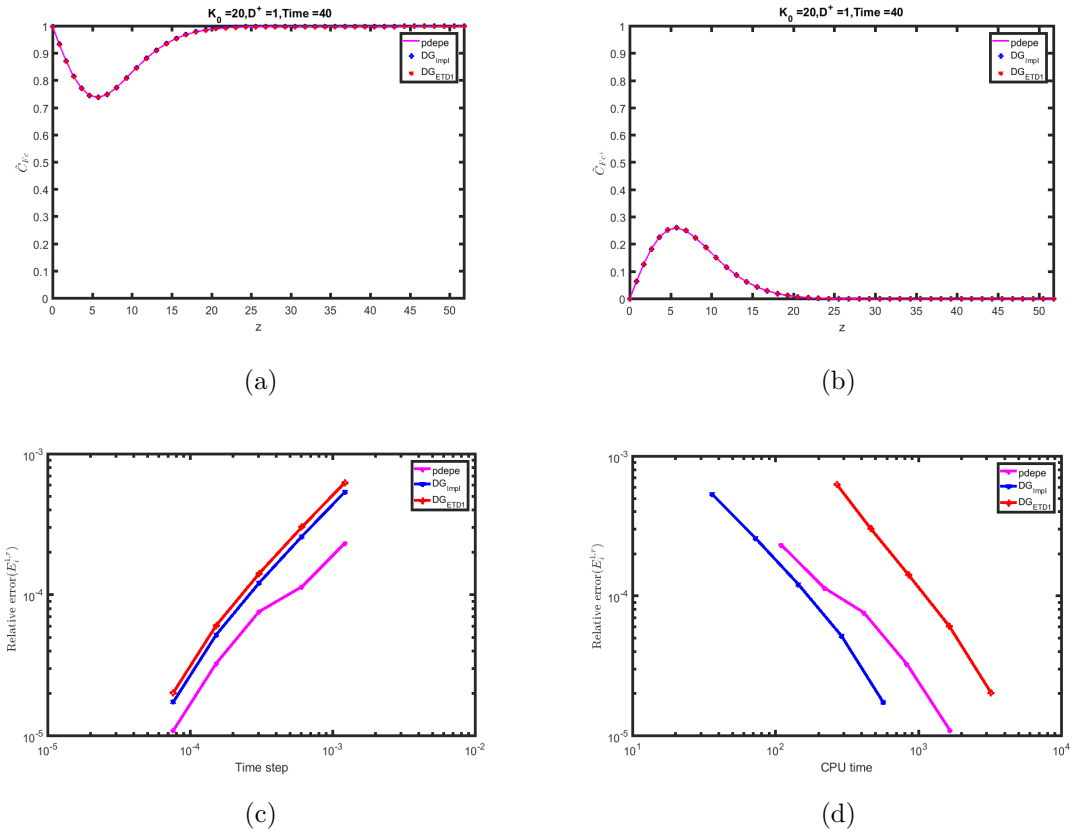


Figure 3.8: **Convergence and the efficiency with respect to the time discretization while computing $\tilde{C}_{\mathbf{Q}}$ and $\tilde{C}_{\mathbf{Q}^+}$ with the solvers pdepe, DG_{Impl} and DG_{ETD_1}.** We respectively plot $\tilde{C}_{\mathbf{Q}}$ and $\tilde{C}_{\mathbf{Q}^+}$ at final time $\tilde{t}_2$ for $\Delta t_0 = 3.8147 \times 10^{-5}$ in (a) and (b). The convergence and the efficienciness in this case are respectively illustrated by plotting $E^{1,r}_i$ vs Δt_i in (c) and $E^{1,r}_i$ vs CPU time in (d) for $i = 1, \cdots, 5$. (c) shows that the solvers *pedpe* , DG_{Impl} and DG_{ETD_1} have the same order of convergence with respect to the time step. (d) shows that DG_{Impl} is more efficient, with respect to the time step, to compute the concentration of the species $\mathbf{Q}$ and $\mathbf{Q}^+$ at the final time $\tilde{t}_2$.

Finally, we examine the convergence and efficiency with respect to the mean of the space step by computing the concentration profile of the species $\mathbf{Q}$ and $\mathbf{Q}^+$ at the final time, using *pedpe*, DG_{Impl} and DG_{ETD_1}, for the each partition Ω_i, for

$i = 1, \cdots, 5$. The partitions Ω_i , for $i = 2, \cdots, 5$ are obtained by splitting each element of Ω_1 into i equidistant elements, for a given partition Ω_1. By considering the concentration profile associated to the finest space step H_5 of the partition Ω_5 as exact concentration, we compute the relative error associated to the mean of the space step, H_i of the partition Ω_i, $i \in \{1, \cdots, 4\}$, as follow

$$E_i^{2,r} = 100 \frac{\sqrt{\sum_{j=\mathbf{Q},\mathbf{Q}^+} \left| C_j^{5,r} - C_j^{i,r} \right|^2_{L^2(\Omega_i)}}}{\sqrt{\sum_{j=\mathbf{Q},\mathbf{Q}^+} \left| C_j^{5,r} \right|^2_{L^2(\Omega_5)}}}, \tag{3.64}$$

for the each solver $r = \texttt{pdepe}, DG_{Impl}, DG_{ETD_1}$. We respectively plot the dimensionless concentrations $\tilde{C}_{\mathbf{Q}}$ and $\tilde{C}_{\mathbf{Q}^+}$ at final time $\tilde{t}_2$ for $H_5 = 0.0582$ in Fig 3.9(a) and Fig 3.9(b). The convergence and the efficiency in this case are respectively illustrated by plotting $E_i^{1,r}$ vs H_i in Fig 3.9(c) and $E_i^{1,r}$ vs CPU time in Fig 3.9(d). Note that in Fig 3.9(c), the relative error $E_i^{2,r}$ decrease with the mean of the space step H_i for all $r = \texttt{pdepe}, DG_{Impl}, DG_{ETD_1}$. It also shows that

$$E_i^{2,\texttt{pdepe}} > E_i^{2,r}, \quad E_i^{2,r} = E_i^{2,l}, \tag{3.65}$$

for all $r, l \in \{DG_{Impl}, DG_{ETD_1}\}$. Therefore the solver DG_{Impl} and DG_{ETD_1} are more accurate than `pdepe` with respect to the space discretization. Fig 3.9(d) shows that for a given relative E^0, we have

$$T^0_{\texttt{pdepe}} > T^0_{DG_{ETD_1}} > T^0_{DG_{Impl}}, \tag{3.66}$$

where T_r^0 is the CPU time spends by the solver $r = \texttt{pdepe}, DG_{Impl}, DG_{ETD_1}$ for the computation of the dimensionless concentration $\tilde{C}_{\mathbf{Q}}$ and $\tilde{C}_{\mathbf{Q}^+}$ such that $E_{i_0}^{2,r} = \mathcal{E}^0$ at the time $\tilde{t} = \tilde{t}_2$. Thus DG_{Impl} is the most efficient solver with respect to the space discretization.

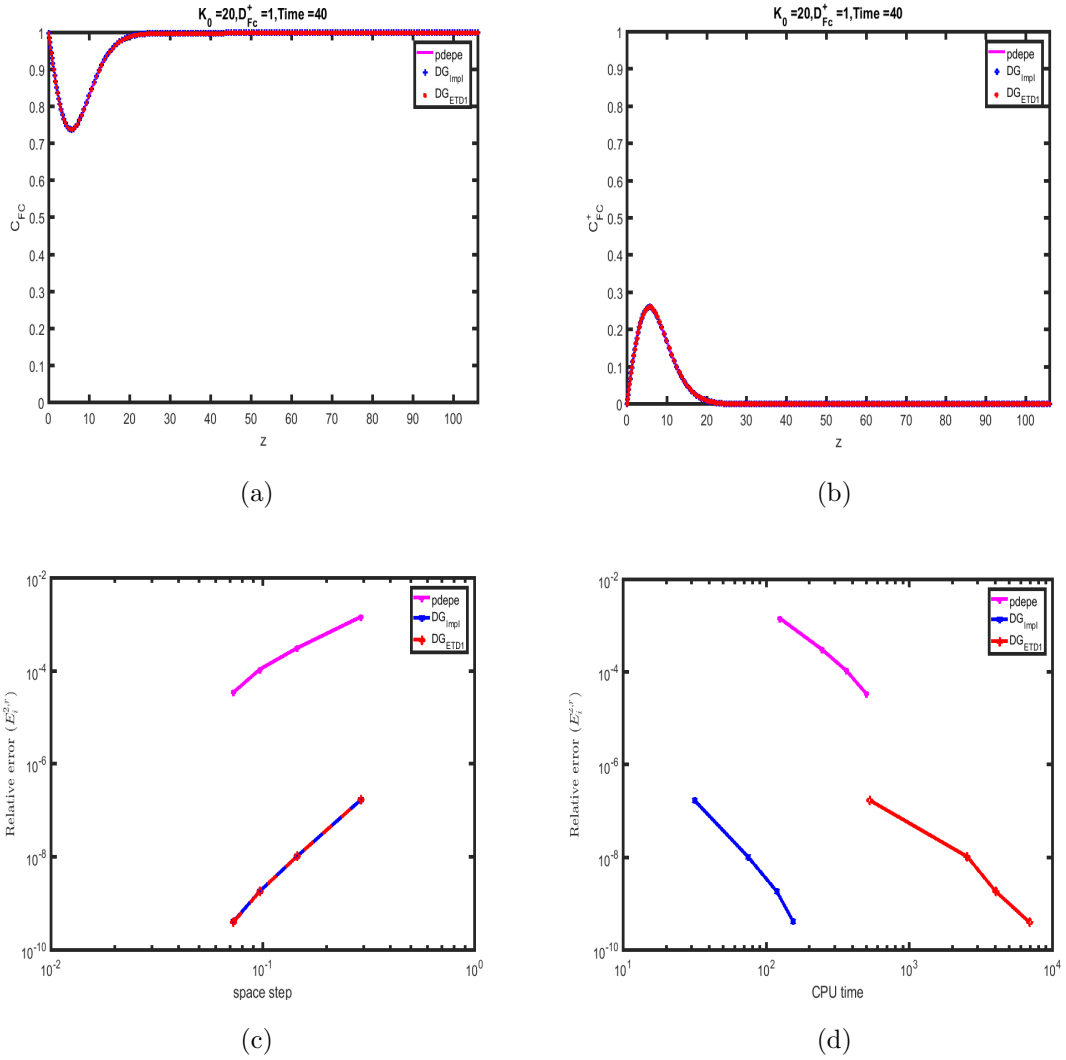


Figure 3.9: **Convergence and the efficiency with respect to the space discretization while computing $\tilde{C}_{\mathbf{Q}}$ and $\tilde{C}_{\mathbf{Q}^+}$ with the solvers** `pdepe`, DG_{Impl} **and** DG_{ETD_1}**.** We respectively plot $\tilde{C}_{\mathbf{Q}}$ and $\tilde{C}_{\mathbf{Q}^+}$ at final time $\tilde{t}_2$ for $H_5 = 0.0582$ in (a) and (b). The convergence and the efficiency in this case are respectively illustrated by plotting $E_i^{1,r}$ vs H_i in (c) and $E_i^{1,r}$ vs CPU time (d). (c) shows that the solvers *pedpe* , DG_{Impl} and DG_{ETD_1} have the same order of convergence with respect to the space step. (d) shows that DG_{Impl} is more efficient, with respect to the space step, to compute the concentration of the species $\mathbf{Q}$ and $\mathbf{Q}^+$ at the final time $\tilde{t}_2$.

Numerical simulation of the dimensionless voltammogram

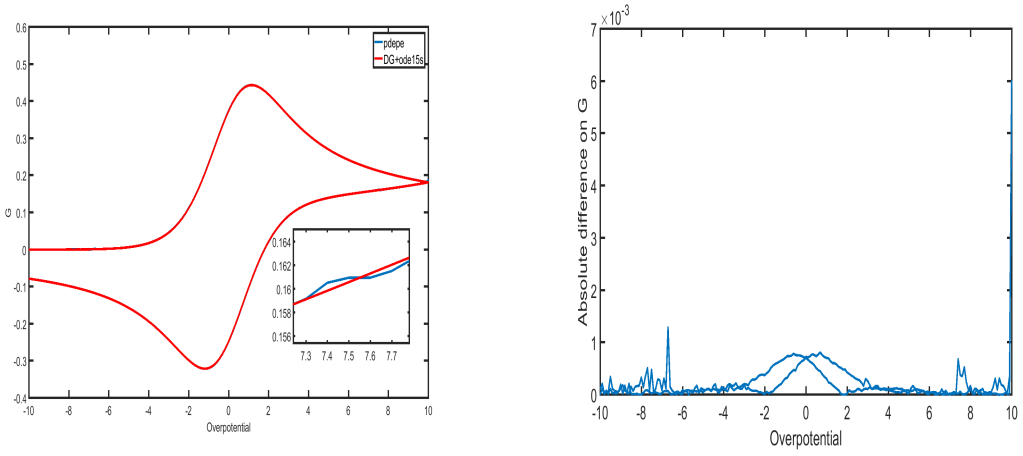
The voltammogram is a finger print of our model and depends only on the concentration of the species $\mathbf{Q}$ and $\mathbf{Q}^+$ at the boundary $z = 0$ and the potential. We now investigate by numerical simulations if DG_{Impl} is still the better solver of the ETO model.

Since Matlab's solver `pdepe` used FE method with *ode*15*s* solver, then in order

to compare space discretization FE method and DG method on the simulation of the volatammogram, we compare the solution obtained using DG method combined with MATLAB's solver $ode15s$ (DG_{ode15s}) with the solution obtained using Matlab's solver `pdepe`. Fig 3.10(a) shows both voltammograms and by zooming, we note that the voltammogram obtained with `pdepe` present some oscillations. We plot in Fig 3.10(b), the absolute value of the difference between both voltammograms. it shows that despite the oscillation of the voltammogram obtained with `pdepe`, both voltammograms are almost the same since we have

$$\left| G_{\texttt{pdepe}}(\tilde{t}) - G_{DG_{ode15s}}(\tilde{t}) \right| < 7 \times 10^{-3},$$

where $G_{\texttt{pdepe}}$ and $G_{DG_{ode15s}}$ respectively represent the current obtained with `pdepe` and DG combined with $ode15s$. We can conclude from the results illustrated in Fig 3.10 that our DG discretisation, compared to the FE method of the solver `pdepe`, is more suitable to compute a more stable voltamogramm (without the oscillations).



(a) (b)

Figure 3.10: **Comparison of MATLAB's solver `pdepe` and the solver DG combined with $ode15s$ for the ETO model with $\tilde{D}_{\mathbf{Q}} = \tilde{D}_{\mathbf{Q}^+} = 1$ and $K_0 = 20$.** In (a) we plot both voltammograms and a the highlight of their portion. we note that the voltammogram obtained with `pdepe` present some oscillations. In (b) we plot the absolute value of the difference between both voltammograms. it shows that despite the oscillation of the voltammogram obtained with `pdepe`, both voltammograms are almost the same

In order to find the most appropriate time discretization method to combine with DG method, to accurately simulate the voltammogram, we investigate the

dimensionless voltammogram using the solver *pedpe*, DG_{Impl} and DG_{ETD_1}. The exponential time differencing method ETD_1 is implemented in three different ways using the functions *expm*, detailed in [126, 5, 120, 169], *phiv* and *phipm*, detailed in [172, 217], to compute $exp(W)X$, where W is a squared matrix and X a vector. Let us introduce the notations $DG_{ETD_1^{expm}}, DG_{ETD_1^{phiv}}$ and $DG_{ETD_1^{phipm}}$ to respectively denote DG combined with ETD_1 solver implemented with $phipm, phiv$ and $phipm$. For different values of the time step Δt, we simulate the dimensionless voltammogram using the solvers *pedpe*, DG_{Impl}, $DG_{ETD_1^{expm}}, DG_{ETD_1^{phiv}}$ and $DG_{ETD_1^{phipm}}$. We respectively plot in Fig 3.11(a), Fig 3.11(b) and Fig 3.11(c) the simulated dimensionless voltammogram of each solver for $\Delta t_1 = 10^{-1}$, $\Delta \tilde{t}_2 = 10^{-2}$ and $\Delta t_3 = 10^{-3}$. we also indicate in Fig 3.11(a), Fig 3.11(b) and Fig 3.11(c) the position of dimensionless peak cathodic current G_{pc}^{20}. The results illustrated in Fig 3.11(a), Fig 3.11(b) and Fig 3.11(c) show that as we the time step tend to zero, the voltammogramm obtain with the solvers *pedpe*, DG_{Impl}, $DG_{ETD_1^{expm}}, DG_{ETD_1^{phiv}}$ and $DG_{ETD_1^{phipm}}$ converge but `pdepe` and DG_{Impl} give a better approximation of the dimensionless peak cathodic current. We associate to each time step $\Delta t_i, i = 1, 2, 3$, the absolute error $E_r^{3,i}$ given by

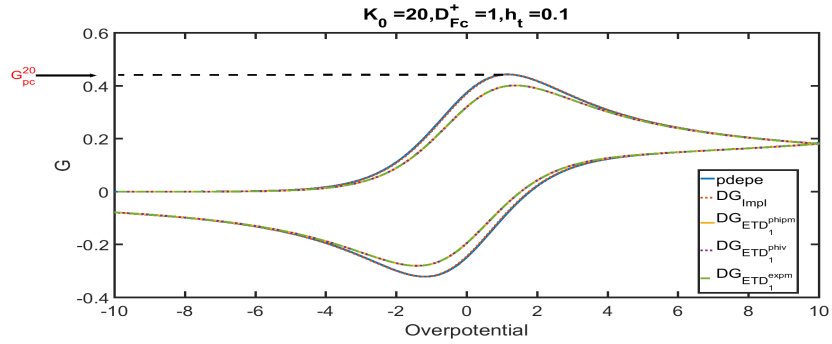
$$E_r^{3,i} = 100\frac{\left|G^i_{\texttt{pdepe}} - G^i_{DG_r}\right|_{L^2(P)}}{\left|G^i_{\texttt{pdepe}}\right|_{L^2(P)}}, \quad \left|G\right|^2_{L^2(P)^2} = \left|\int_{FS} G^2 dP\right| + \left|\int_{RS} G^2 dP\right|,$$

where FS and RS denote respectively the forward sweep and the reverse sweep, P the overpotential and $G^i_{DG_r}$ represent the dimensionless current obtained with the time solver $r = Impl, ETD_1^{expm}, ETD_1^{phiv}, ETD_1^{phipm}$ for the time step Δt_i. We record in Tab 3.1, the relative error $E_r^{3,i}$ associated to the time solver $r = Impl, ETD_1^{expm}, ETD_1^{phiv}, ETD_1^{phipm}$ for a given time step $\Delta t_i, i = 1, 2, 3$. Note that the relative error decrease with the time step; and for a given time step Δt_i, we have

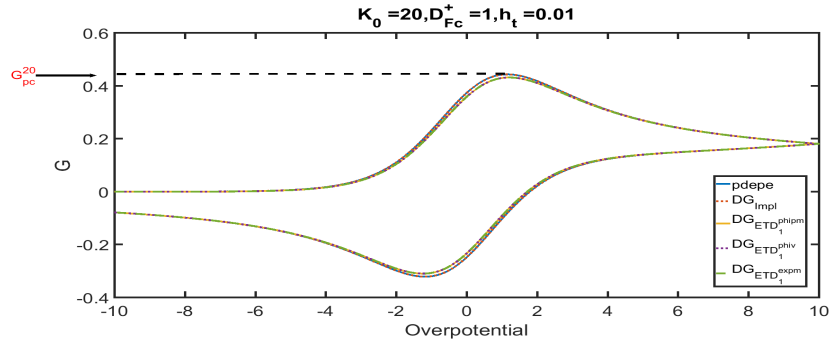
$$E^{3,i}_{Impl} < E^{3,i}_r, \quad E^{3,i}_r = E^{3,i}_l$$

for all $l, r = ETD_1^{expm}, ETD_1^{phiv}, ETD_1^{phipm}$. Therefore the we can conclude that the

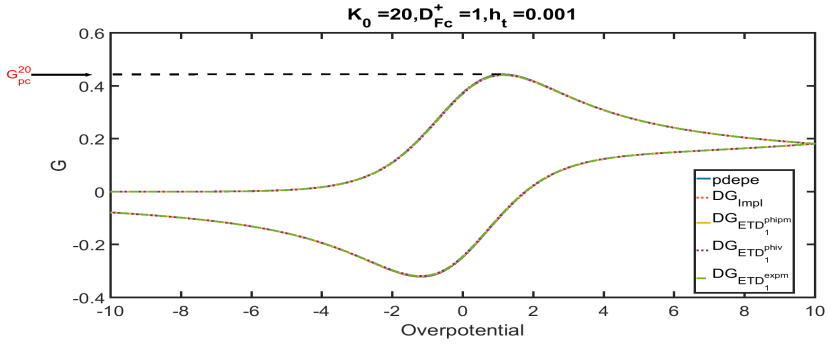
voltammogram simulated with ETD_1 doesn't depend on the function used for the computation $exp(W)X$. It also show that the Implicit Euler method gives a more accurate voltammogram compared to the exponential time differencing method for a given time step.



(a)



(b)



(c)

Figure 3.11: **Dimensionless voltammogram simulated with** *pedpe*, DG_{Impl}, $DG_{ETD_1^{expm}}$, $DG_{ETD_1^{phiv}}$ **and** $DG_{ETD_1^{phipm}}$ **for different time step.** We respectively plot in (a), (b) and (c) the simulated dimensionless voltammogram of each solver for $\Delta t_1 = 10^{-1}$, $\Delta \tilde{t}_2 = 10^{-2}$ and $\Delta t_3 = 10^{-3}$. we also indicate in (a),(b) and (c) the position of dimensionless peak cathodic current G_{pc}^{20}. This figure shows that all the solvers are converging with respect to the time discretisation. It also shows that `pdepe` and DG_{Impl} give a better approximation of the dimensionless peak cathodic current.

	$E^{3,i}_{Impl}$	$E^{3,i}_{ETD_1^{phipm}}$	$E^{3,i}_{ETD_1^{phiv}}$	$E^{3,i}_{ETD_1^{expm}}$
$\Delta t_1 = 10^{-1}$	0.6624%	9.9122%	9.9122%	9.9122%
$\Delta t_2 = 10^{-2}$	0.0767%	2.7925%	2.7925%	2.7925%
$\Delta t_3 = 10^{-3}$	0.0266%	0.5985%	0.5985%	0.5985%

Table 3.1: **Relative error on the computation of the current**. We record in this table, the relative error $E^{3,i}_r$ associated to the time solver $r = Impl, ETD_1^{expm}, ETD_1^{phiv}, ETD_1^{phipm}$ for a given time step $\Delta t_i, i = 1, 2, 3$. It shows that the relative error decrease with the time step. It also show that for a given time step Δt_i, we have $E^{3,i}_{Impl} < E^{3,i}_r$ for all $r = ETD_1^{expm}, ETD_1^{phiv}, ETD_1^{phipm}$.

To find out if the bad performance of exponential time differencing method is related to the maximum degree, k_j, of the local basis function of the finite space $\mathcal{P}^{k_j}(I_j)$, we simulate the dimensionless voltammogram for different values of k_j using DG_{Impl} and DG_{ETD_1}. For all $k \in \{1, 2, 3\}$ and $h_t = 10^{-1}$, we respectively plot in Fig 3.12(a) and Fig 3.12(b), the dimensionless voltammogram simulated with DG_{Impl} and DG_{ETD_1} with $k_j = k, \forall j \in \{1, \cdots, n\}$. In both Fig 3.12(a) and Fig 3.12(b), we also plot the dimensionless voltammogram simulated with `pdepe` and indicate the position of dimensionless peak cathodic current G^{20}_{pc}. For all $k = 1, 2, 3$ we compute the relative error $E^{r,k}_{G_{pc}}$ on the simulated dimensionless peak current and the relative error $E^{l,k}_G$ on the simulated dimensionless current as follow

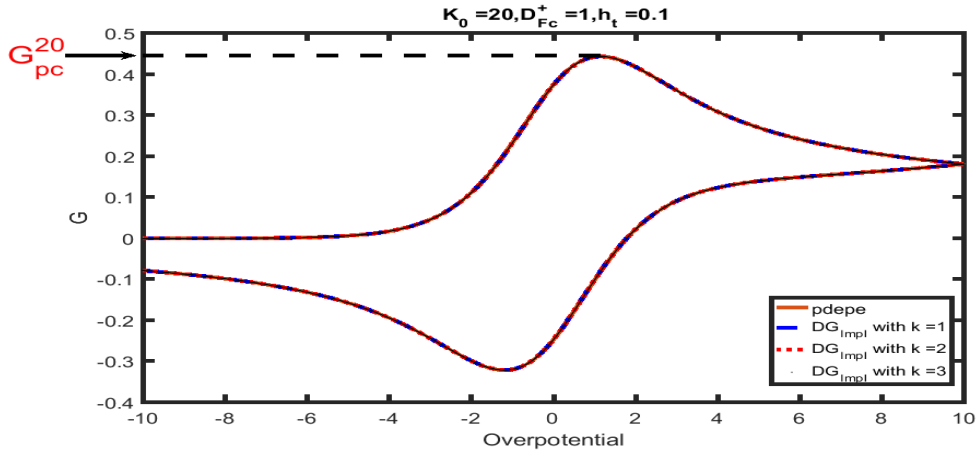
$$E^{r,k}_{G_{pc}} = 100\frac{\left|G^{r,k}_{pc} - G^{20}_{pc}\right|}{\left|G^{20}_{pc}\right|}, \quad E^{l,k}_G = 100\frac{\left|G^{l}_{k} - G^{\texttt{pdepe}}_{k}\right|_{L^2(P)}}{\left|G^{\texttt{pdepe}}_{k}\right|_{L^2(P)}}, \tag{3.67}$$

where $G^{r,k}_{pc}$ and G^{l}_{k} are respectively the simulated dimensionless peak cathodic current with the solver $r \in \{\texttt{pdepe}, DG_{Impl}, DG_{ETD_1}\}$ and the dimensionless current simulated with the solver $l \in \{DG_{Impl}, DG_{ETD_1}\}$. Tab 3.2 gives the values of $E^{r,k}_{G_{pc}}$ and $E^{l,k}_G$ for all $r \in \{\texttt{pdepe}, DG_{Impl}, DG_{ETD_1}\}$, $l \in \{DG_{Impl}, DG_{ETD_1}\}$ and $k \in \{1, 2, 3\}$. Note that in Tab 3.2, we have for all $k \in \{1, 2, 3\}$

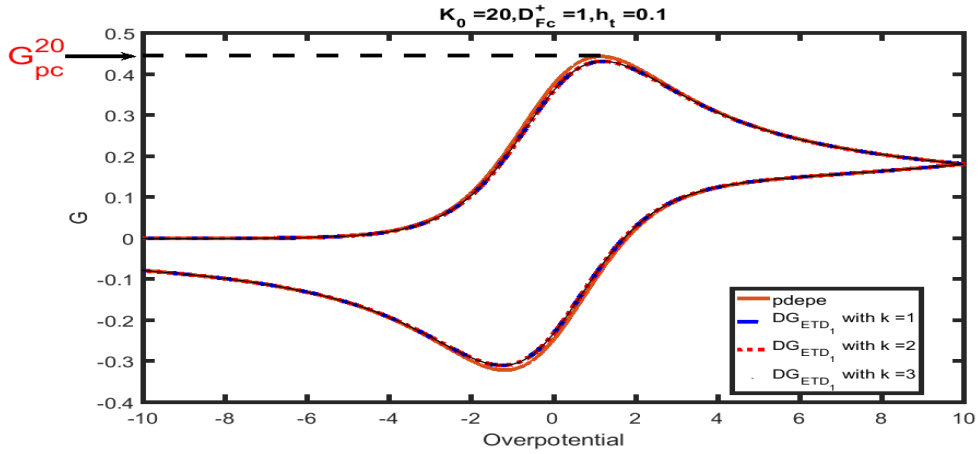
$$E^{\texttt{pdepe},k}_{G_{pc}} \approx E^{DG_{Impl},k}_{G_{pc}} < E^{DG_{ETD_1},k}_{G_{pc}}, \quad E^{DG_{Impl},k}_{G} < E^{DG_{ETD_1},k}_{G}. \tag{3.68}$$

We can conclude from Fig 3.11 and (3.68), we can conclude that DG_{Impl} is better

solver compared DG_{ETD_1} while computing either the dimensionless peak cathodic current or the dimensionless current. Tab 3.2 also shows that the relative errors $E^{r,k}_{G_{pc}}$ and $E^{l,k}_{G}$ are practically independent of the k. Then we can conclude that the bad performance of the ETD_1 doesn't depend on the value of k.



(a)



(b)

Figure 3.12: **Simulated voltammogram for different maximum degree k_j of the basis function**. For all $k \in \{1, 2, 3\}$ and $\Delta t = 10^{-1}$, We respectively plot in (a) and (b), the dimensionless voltammogram simulated with DG_{Impl} and DG_{ETD_1} with $k_j = k, \forall j \in \{1, \cdots, n\}$. In both (a) and (b), we also plot the dimensionless voltammogram simulated with `pdepe` and indicate the position of dimensionless peak cathodic current $G^2_{pc}0$. Note that `pdepe` and DG_{Impl}, give better approximation of dinmensionless peak cathodic current, compared to DG_{ETD_1}.

	$E^{r,k}_{G_{pc}}(\%)$			$E^{l,k}_{G}(\%)$	
	$r=$ pdepe	$r=DG_{Impl}$	$r=DG_{ETD_1}$	$l=DG_{Impl}$	$l=DG_{ETD_1}$
k=1	0.422	0.4406	9.8154	0.6596	9.9220
k=2	0.422	0.4483	9.8078	0.6653	9.9125
k=3	0.422	0.4407	9.8060	0.6624	9.9122

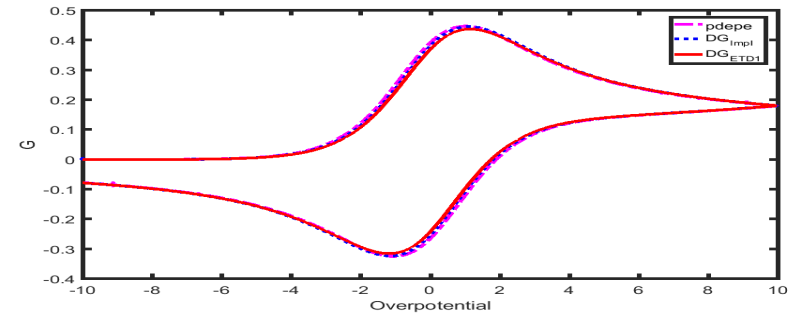
Table 3.2: **Relative error on the simulated dimensionless current and peak cathodic current for different maximum degree k_j of the basis function**. We record in this table the relative error $E^{r,k}_{G_{pc}}$ on the simulated dimensionless peak current and the relative error $E^{l,k}_{G}$ on the simulated dimensionless current for all $k_j=k$, $r\in\{\texttt{pdepe}, DG_{Impl}, DG_{ETD_1}\}$, $l\in\{DG_{Impl}, DG_{ETD_1}\}$ and $k\in\{1,2,3\}$.

Let us now examine the convergence and the efficiency of the solvers *pedpe*, DG_{Impl} and DG_{ETD_1} with respect to the time and space step, while computing the dimensionless current. To do so, we firstly simulate the dimensionless voltammogram using the solvers *pedpe*, DG_{Impl} and DG_{ETD_1} for all time step $\Delta t_i = 2^i \times \Delta t_0$, for $i \in \{0,\cdots,5\}$. By considering the dimensionless current associated to the finest time step Δt_0 as the exact dimensionless current, we compute the relative error associated to the time step $\Delta t_i = 2^i \times \Delta t_0$, for $i \in \{1,\cdots,5\}$ as follow

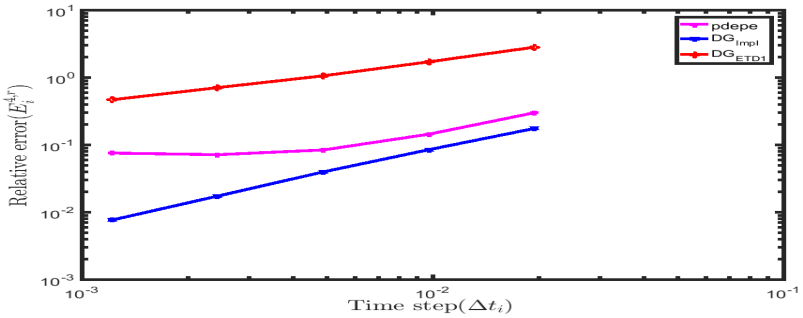
$$E_i^{4,r} = 100\frac{\left|G^{0,r}-G^{i,r}\right|^2_{L^2(P)}}{\left|G^{0,r}\right|^2_{L^2(P)}}, \tag{3.69}$$

where P is the overpotential and $G^{i,r}$ is the dimensionless current simulated with the solver $r=\texttt{pdepe}, DG_{Impl}, DG_{ETD_1}$ for Δt_i. In Fig 3.13(a) we plot the dimensionless voltammogram simulated by pdepe, DG_{Impl} and DG_{ETD_1} with $\Delta t_0 = 3.8147\times 10^{-5}$. The convergence and the efficiency in this case are respectively illustrated by plotting $E_i^{4,r}$ vs Δt_i in Fig 3.13(b) and $E_i^{4,r}$ vs CPU time in Fig 3.13(c) for $i=1,\cdots,5$. Note that in Fig 3.13(b), the relative error $E_i^{4,r}$ tend to zeros with the time step for all $r=\texttt{pdepe}, DG_{Impl}, DG_{ETD_1}$, meaning the three solvers converge with respect to the time step while simulating the voltammogram. Fig 3.13(c) shows that for a given value E_0, DG_{Impl} compared to pdepe, DG_{ETD_1} will spend less time to simulate the dimensionless voltammogram with a relative error equal E_0. Thus in

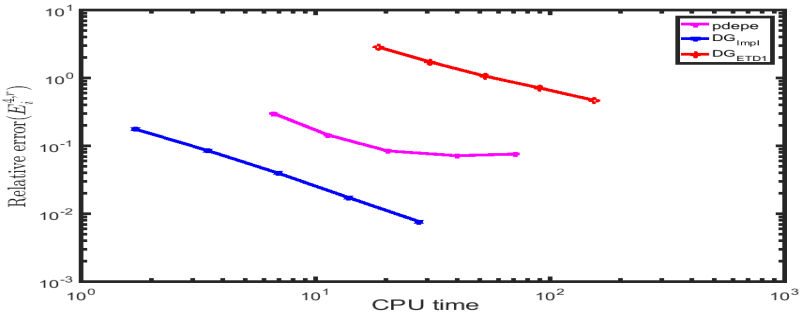
this case, DG_{Impl} is more efficient than `pdepe`, DG_{ETD_1} to simulate the dimensionless voltammogram. The curve of `pdepe` in Fig 3.13(b) and Fig 3.13(c) present a plateau, due to the fact that for a given time step Δt, `pdepe` use an adaptive time solver with time step less or equal to Δt.



(a)



(b)



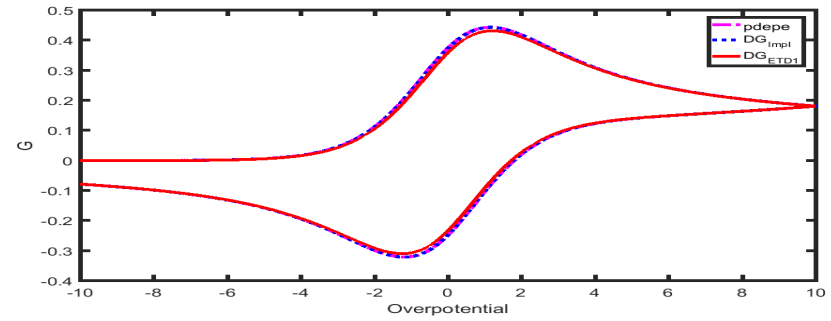
(c)

Figure 3.13: **Convergence and the efficiency with respect to the time step while simulating the voltammogram**. In (a) we plot the dimensionless voltammogram simulated by `pdepe`, DG_{Impl} and DG_{ETD_1} with $\Delta t_0 = 6.1036 \times 10^{-4}$. The convergence and the efficiency in this case are respectively illustrated by plotting $E_i^{4,r}$ vs Δt_i in (b) and $E_i^{4,r}$ vs CPU time in (c) for $i = 1, \cdots, 5$. Note that in (b), the relative error $E_i^{4,r}$ tend to zeros with the time step for all $r =$ `pdepe`, DG_{Impl}, DG_{ETD_1}. Meaning the three solvers converge with respect to the time step while simulating the voltammogram. (c) shows that for a given value E_0, DG_{Impl} compared to `pdepe`, DG_{ETD_1} will spend less time to simulate the dimensionless voltammogram with a relative error equal E_0. Thus DG_{Impl} is more efficient.

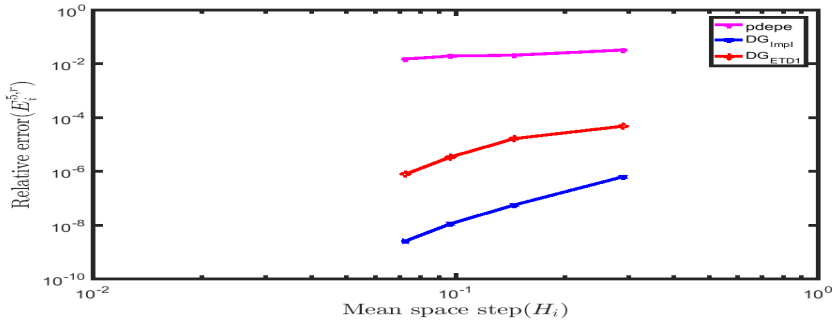
We finally simulate the dimensionless voltammogram using the solvers *pedpe*, DG_{Impl} and DG_{ETD_1}, for all partition $\Omega_i,\ i = 1, \cdots, 5$. The partitions Ω_i , for $i = 2, \cdots, 5$ are obtained by splitting each element of Ω_1 into i equidistant elements, for a given partition Ω_1. By considering the dimensionless voltammogram associated to the finest mean space step H_5 of the partition Ω_5 as exact dimensionless voltammogram, we compute the relative error $E_i^{5,r}$ associated to the mean space step H_i of the partition $\Omega_i,\ i \in \{1, \cdots, 4\}$ as follows

$$E_i^{5,r} = 100 \frac{\left| G^{0,r} - G^{i,r} \right|^2_{L^2(P)}}{\left| G^{0,r} \right|^2_{L^2(P)}}, \tag{3.70}$$

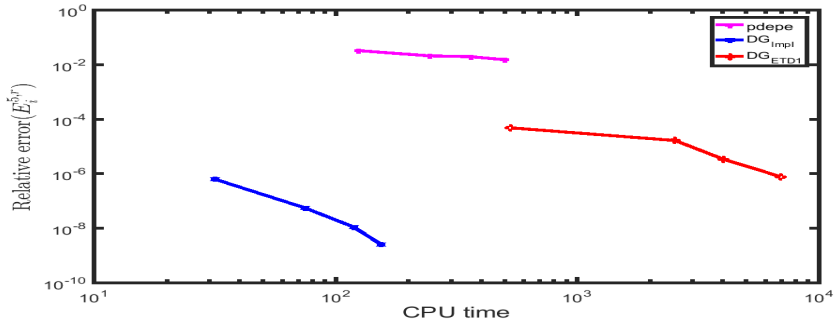
where P is the overpotential and $G^{i,r}$ is the dimensionless current simulated with the solver $r = \texttt{pdepe}, DG_{Impl}, DG_{ETD_1}$ for Ω_i. In Fig 3.14(a) we plot the dimensionless voltammogram simulated by `pdepe`, DG_{Impl} and DG_{ETD_1} with $H_5 = 0.0582$. The convergence and the efficiency in this case are respectively illustrated by plotting $E_i^{5,r}$ vs H_i in Fig 3.14(b) and $E_i^{5,r}$ vs CPU time in Fig 3.14(c) for $i = 1, \cdots, 5$. Note that in Fig 3.14(b), the relative error $E_i^{5,r}$ tend to zeros with the mean space step for all $r = \texttt{pdepe}, DG_{Impl}, DG_{ETD_1}$. Then the three solvers converge with respect to the mean space step while simulating the voltammogram. Fig 3.14(c) shows that for a given value E_0, DG_{Impl} compared to $\texttt{pdepe}, DG_{ETD_1}$ will spend less time to simulate the dimensionless voltammogram with a relative error equal E_0. Thus in this case, DG_{Impl} is more efficient than $\texttt{pdepe}, DG_{ETD_1}$ to simulate the dimensionless voltammogram.



(a)



(b)



(c)

Figure 3.14: **Convergence and the efficiency with respect to the mean of the space steps while simulating the voltammogram :** In (a) we plot the dimensionless voltammogram simulated by `pdepe`, DG_{Impl} and DG_{ETD_1} with $H_5 = 0.0582$. The convergence and the efficiency in this case are respectively illustrated by plotting $E_i^{5,r}$ vs H_i in (b) and $E_i^{5,r}$ vs CPU time in (c) for $i = 1, \cdots, 5$. Note that in (b), the relative error $E_i^{5,r}$ tend to zeros with the mean space step for all $r = \texttt{pdepe}, DG_{Impl}, DG_{ETD_1}$. Then the three solvers converge with respect to the mean space step while simulating the voltammogram. (c) shows that for a given value E_0, DG_{Impl} compared to `pdepe`, DG_{ETD_1} will spend less time to simulate the dimensionless voltammogram with a relative error equal E_0. Thus DG_{Impl} is more efficient.

Another way to strengthen the effectiveness of our solver $\mathrm{DG_{Impl}}$ is to verify the Property 3.1 by simulating the dimensionless peak cathodic current for several values of the parameters $\tilde{D}_{Fc^+}$ and K_0, using $\mathrm{DG_{Impl}}$. For the default dimensionless electron

transfer rate K_0 coupled with several dimensionless diffusion coefficients $\tilde{D}^i_{Fc^+}, i \in \{1, \cdots, 7\}$ as illustrated by Tab 3.3, we simulate the dimensionless current $G^i_{\tilde{D}_{Fc^+}}$ using the solver $\mathrm{DG_{Impl}}$. We then plot in Fig 3.15, the dimensionless currents $G^i_{\tilde{D}_{Fc^+}}$ as function of the overpotential for all $i \in \{1, \cdots, 7\}$. As expected, Fig 3.15 shows that the dimensionless peak cathodic current does not depend on the dimensionless diffusion coefficients $\tilde{D}_{Fc^+}$.

i	1	2	3	4	5	6	7
$D^i_{Fc^+}$	1	2	4	6	8	10	20

Table 3.3: Settings to investigate the independence of the dimensionless peak cathodic current with respect the dimensionless diffusion coefficients of the ferrocenium.

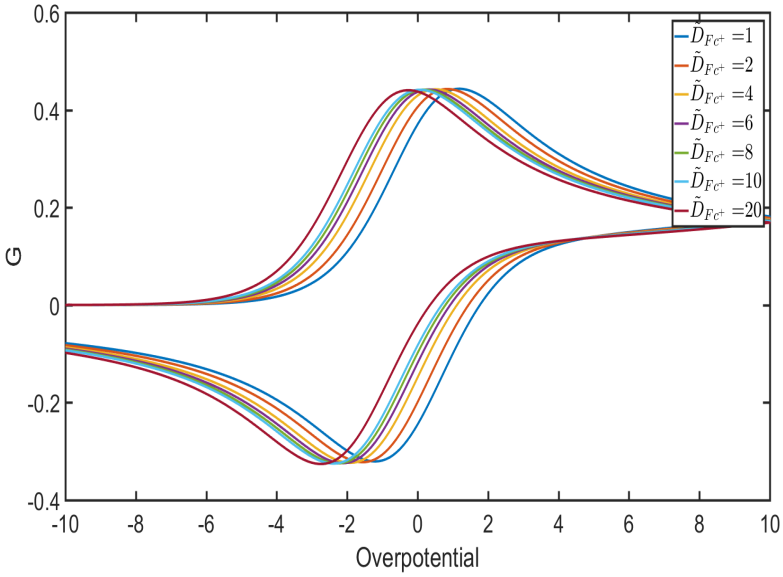


Figure 3.15: Variation of the voltammogram with respect to the dimensionless diffusion coefficients of the ferrocenium. This shows that the dimensionless peak cathodic current is independent of the dimensionless diffusion coefficients of the ferrocenium.

For the default dimensionless diffusion coefficients D_{Fc^+} coupled with several dimensionless electron transfer rate $K^i_0, i \in \{1, \cdots, 7\}$, as illustrated by Tab 3.4, we simulate the dimensionless current $G^i_{K_0}$ using the solver $\mathrm{DG_{Impl}}$. We then plot in Fig 3.16, the dimensionless currents $G^i_{K_0}$ as function of the overpotential for all $i \in \{1, \cdots, 7\}$. As expected, Fig 3.16 shows that the dimensionless peak cathodic current increases to a certain limit as we increase the dimensionless electron transfer rate K_0.

i	1	2	3	4	5	6	7
K_0^i	0.5	2	5	10	20	30	50

Table 3.4: Settings to investigate the variation of the dimensionless peak cathodic current with respect the dimensionless electron transfer rate.

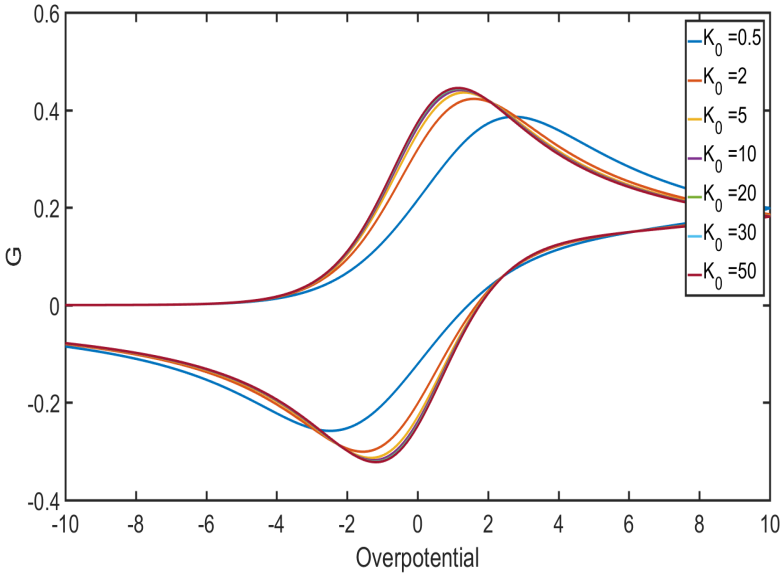
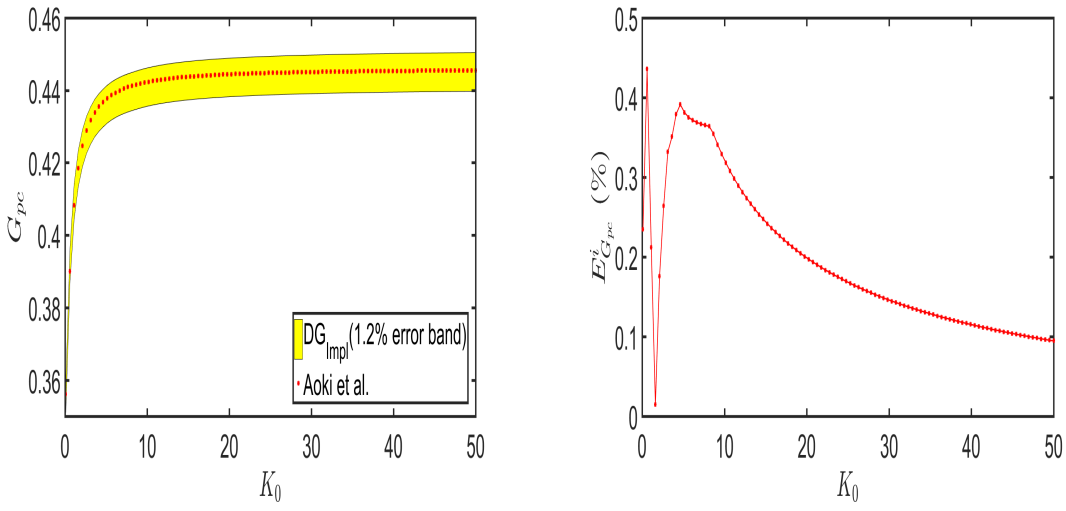


Figure 3.16: Variation of the voltammogram with respect to the dimensionless electron transfer rate. This shows that the dimensionless peak cathodic current increases to a certain limit as we increase the dimensionless electron transfer rate.

For several dimensionless electron transfer rate, $K_0^i \in [0.1, 50]$ and transfer coefficient $\alpha = 0.5$, we generate the dimensionless peak cathodic currents $\tilde{G}_{pc}^i$ and G_{pc}^i using respectively our solver $\mathrm{DG}_{\mathrm{Impl}}$ and the approximation formula (3.16), and compute the relative error $E_{G_{pc}}^i$ as follow

$$K_0^i = 0.1 + i \times \frac{50 - 0.1}{100}, \quad E_{G_{pc}}^i = 100 \frac{|\, \tilde{G}_{pc}^i - G_{pc}^i \,|}{\tilde{G}_{pc}^i}, \quad \forall i \in \{0, 1, \cdots, 100\}.$$

We plot in Fig 3.17 (a) the dimensionless peak cathodic current G_{pc}^i and 1.2% relative error band around dimensionless peak cathodic current $\tilde{G}_{pc}^i$, as a function of the dimensionless electron transfer rate, K_0^i. We plot in Fig 3.17 (b) the relative error $E_{G_{pc}}^i$ against the dimensionless electron transfer rate, K_0^i. As expect, Fig 3.17 (a) shows that the simulated peak cathodic current G_{pc}^i lays in the shaded error band for all $i \in \{0, 1, \cdots, 100\}$, while Fig 3.17 (b) shows that the relative $E_{G_{pc}}^i < 0.5\%$ for all $i \in \{0, 1, \cdots, 100\}$, meaning the Property 3.1 is satisfied.



(a) (b)

Figure 3.17: **Comparison of the simulated and the analytic approximated peack cathodic current.** In (a), we plot the dimensionless peak cathodic obtained with the Aoki et al. approximation formula and the 1.2% relative error band around the $\mathrm{DG_{Impl}}$ simulated dimensionless peak cathodic against the dimensionless electron transfer rate K_0, for $\alpha = 0.5$. In (b), we plot the relative error between the simulated and the approximated dinmensionless peak cathodic as a function of K_0.

3.3 DG for electro catalytic model

We now introduce an electro catalytic reaction (EC') model of four species defined by the following chemical reactions

$$\mathbf{Q} \underset{k_b}{\overset{k_f}{\rightleftharpoons}} \mathbf{Q}^+ + e^-, \quad B + 2\mathbf{Q}^+ \xrightarrow{k_r} 2\mathbf{Q}, \quad A \underset{k_A^-}{\overset{k_A}{\rightleftharpoons}} B. \tag{3.71}$$

The EC' model can then be interpreted as electron transfer followed by a catalytic reaction and an equilibrium reaction. We assume that the species A and B only participate in the homogeneous reactions. Schematically, the EC' mechanism is represented in Fig 3.18.

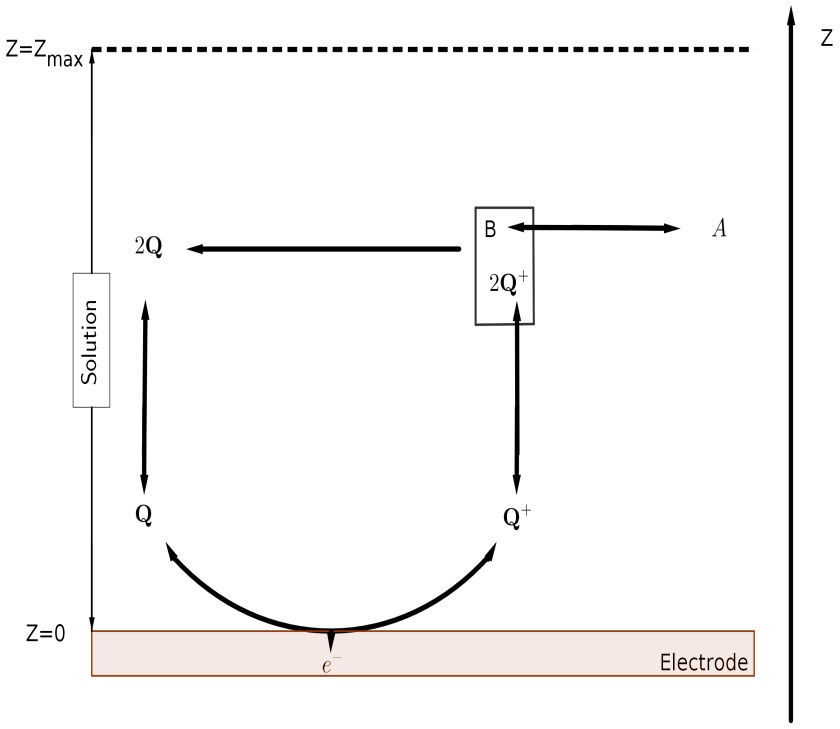


Figure 3.18: **Electron transfer followed by a catalytic and an equilibrium reactions experiment**.

3.3.1 Governing equations

In one dimension the electron transfer followed by a catalytic reaction (3.71), is described by the following system of diffusion reaction PDEs

$$\partial_t C_{\mathbf{Q}} = \partial_Z(D_{\mathbf{Q}}\partial_Z D_{\mathbf{Q}}) + 2k_r C_{\mathbf{Q}^+} C_B, \tag{3.72}$$

$$\partial_t C_{\mathbf{Q}^+} = \partial_Z(D_{\mathbf{Q}^+}\partial_Z C_{\mathbf{Q}^+}) - 2k_r C_{\mathbf{Q}^+} C_B, \tag{3.73}$$

$$\partial_t C_B = \partial_Z(D_B\partial_Z C_B) - k_r C_{\mathbf{Q}^+} C_B + k_A(C_A - C_B), \tag{3.74}$$

$$\partial_t C_A = \partial_Z(D_A\partial_Z C_A) - k_A(C_A - C_B), \tag{3.75}$$

where k_r is the electro catalytic rate constant, k_A is the equilibrium rate constant, the diffusion coefficient of the species $A, B, \mathbf{Q}$ and $\mathbf{Q}^+$ are respectively denoted $D_A, D_B, D_{\mathbf{Q}}$ and $D_{\mathbf{Q}^+}$. The boundary conditions are given by (3.5) to (3.7) for the species $\mathbf{Q}$ and $\mathbf{Q}^+$ involved in the ETO model. Since we assume that the species A and B only participate in the homogeneous reactions, then they are subject to the no flux boundary conditions,

$$\partial_Z C_j\Big|_{Z=Z_{max}} = \partial_Z C_j\Big|_{Z=0} = 0, \qquad j \in \{A, B\}. \tag{3.76}$$

Initial conditions are given by (3.8) for the species involved in ETO model and by (3.77), which represents the ionization of the species A.

$$C_A\Big|_{t=0} = C_B\Big|_{t=0} = \frac{1}{2} C^0_{A^{total}}, \tag{3.77}$$

where $C^0_{A^{total}}$ represent the total concentration of the specie A.

Note from the governing equations that the voltammogram of EC' model is equivalent to the voltammogram of the ETO model when the electro catalytic rate constant is equal to zero ($k_r = 0$).

Dimensionless governing equations

For consistency in the analyses and to make better comparisons to the ETO model, this electron transfer followed by a catalytic reaction model, from (3.72) to (3.77), will also be examined in dimensionless form. Therefore, in addition to the dimensionless parameters used for the electron transfer model in Section 3.2.1, the dimensionless electro catalytic rate constant, K_r, and the dimensionless equilibrium rate constant, K_A, are defined as follow

$$K_r = k_r \tau C_m, \qquad K_A = k_A \tau. \tag{3.78}$$

The dimensionless concentration profile of the species involved in the electron transfer followed by a catalytic reaction model (3.71) follow the system of PDEs

$$\partial_{\tilde{t}} \tilde{C}_{\mathbf{Q}} = \partial_z(\tilde{D}_{\mathbf{Q}} \partial_z \tilde{C}_{\mathbf{Q}}) + 2K_r \tilde{C}_{\mathbf{Q}^+} \tilde{C}_B, \tag{3.79}$$

$$\partial_{\tilde{t}} \tilde{C}_{\mathbf{Q}^+} = \partial_z(\tilde{D}_{\mathbf{Q}^+} \partial_z \tilde{C}_{\mathbf{Q}^+}) - 2K_r \tilde{C}_{\mathbf{Q}^+} \tilde{C}_B, \tag{3.80}$$

$$\partial_{\tilde{t}} \tilde{C}_B = \partial_z(\tilde{D}_B \partial_z \tilde{C}_B) - K_r \tilde{C}_{\mathbf{Q}^+} \tilde{C}_B + K_A(\tilde{C}_A - \tilde{C}_B), \tag{3.81}$$

$$\partial_{\tilde{t}} \tilde{C}_A = \partial_z(\tilde{D}_A \partial_z \tilde{C}_A) - K_A(\tilde{C}_A - \tilde{C}_B), \tag{3.82}$$

subject to the dimensionless boundary conditions (3.19) to (3.21) presented for the

ETO model with the additional dimensionless boundary conditions

$$\partial_z \tilde{C}_j\Big|_{z=z_{max}} = \partial_z \tilde{C}_j\Big|_{z=0} = 0, \qquad j \in \{A, B\}, \tag{3.83}$$

and the dimensionless initial conditions (3.22) presented for ETO model with the additional dimensionless initial conditions (3.84)

$$\tilde{C}_A\Big|_{\tilde{t}=0} = \tilde{C}_B\Big|_{\tilde{t}=0} = \frac{1}{2}\tilde{C}^0_{A^{total}}. \tag{3.84}$$

3.3.2 DG discretisation of the EC' model

To obtain the DG formulation of the EC', we use the same methodology as for the DG formulation of the ETO, described in Section 3.2.2. We apply the DG method to each equation of the dimensionless governing equations of EC' (from (3.79) to (3.82)) and we obtain the system of ODEs

$$\mathbb{M} d_{\tilde{t}} X + (\underbrace{\mathfrak{S}_0 + \mathfrak{S}_1(t) + \mathfrak{S}_2}_{\mathfrak{S}(\tilde{t})})X = N(X, K_r), \tag{3.85}$$

where the vector valued X is the coupled vector of the component of the dimensionless concentration of the specie $\mathbf{Q}(\alpha = (\alpha_r^j))$, of the specie $\mathbf{Q}^+(\alpha^+ = (\alpha_r^{+,j}))$, of the specie $B(\beta^- = (\beta_r^{-,j}))$ and the specie $A(\beta = (\beta_r^j))$ in the finite space V_h

$$X = (\alpha, \alpha^+, \beta^-, \beta)^T.$$

The associated mass matrix $\mathbb{M}$ and stiffness matrix $\mathfrak{S}(\tilde{t})$ can be assembled, for $\tilde{D}_{\mathbf{Q}} = 1, \tilde{D}_{\mathbf{Q}^+} = D^+, \tilde{D}_B = D^-$ and $\tilde{D}_A = D$ as follows

$$\mathbb{M} = \begin{pmatrix} M & 0 & 0 & 0 \\ 0 & M & 0 & 0 \\ 0 & 0 & M & 0 \\ 0 & 0 & 0 & M \end{pmatrix}, \qquad \mathfrak{S}_0 = \left(\begin{array}{c|c|c|c} L_0^{s,\sigma_0} & 0 & 0 & 0 \\ \hline 0 & D^+L_0^{s,\sigma_0} & 0 & 0 \\ \hline 0 & 0 & D^-L_0^{s,\sigma_0} & 0 \\ \hline 0 & 0 & 0 & DL_0^{s,\sigma_0} \end{array}\right),$$

$$\mathfrak{S}_2 = K_A \left(\begin{array}{c|c|c|c} 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ \hline 0 & 0 & M & -M \\ \hline 0 & 0 & -M & M \end{array}\right), \qquad \mathfrak{S}_1(t) = \left(\begin{array}{c|c|c|c} K_f(t)L1 & -K_b(t)L1 & 0 & 0 \\ \hline -K_f(t)L1 & K_b(t)L1 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \end{array}\right),$$

where the block matrices M, L_0^{s,σ_0} and $L1$ are defined while assembling the mass and stiffness matrices associated to the DG formulation of the dimensionless governing equations of ETO model in Section 3.2.2. So the mass matrix $\mathbb{M}$ is then the $4n_T \times 4n_T$ identity matrix. The source term is given by

$$N(X, K_r)^T = K_r(2N_1(X), -2N_1(X), -N_1(X), 0),$$

where N_1 is a $1 \times n_T$ vector defined as follows

$$N_1(X) = (N_{1,r}^j)_{j=1,\cdots,n}^{r=0,\cdots,k_j}, \quad N_{1,r}^j = \int_\Omega \tilde{C}_{\mathbf{Q}^+}\tilde{C}_B\Phi_r^j(z)dz. \tag{3.86}$$

3.3.3 Computation of the concentrations and the current

The system of PDEs (3.79) - (3.82) subject to the boundary conditions (3.19) - (3.21) and (3.83) that describe EC' model, has now been transformed to the system of time dependent ODEs (3.85). The dimensionless initial conditions (3.22) and (3.84) associated to the EC' model is equivalent to

$$X\Big|_{\tilde{t}=0} = (\alpha_0, \alpha_0^+, \beta_0^-, \beta_0)^T, \tag{3.87}$$

where the entries of the vectors α_0 and α_0^+ respectively associated to the species $\mathbf{Q}$ and $\mathbf{Q}^+$, are given by (3.55) while the entries of the vectors β_0^- and β_0 respectively associated to the species A and B are given by

$$\beta_0 = \beta_0^-, \quad \beta_{0,r}^{-,i} = \begin{cases} \frac{1}{2}\tilde{C}^0_{A^{total}}\sqrt{h_i} & \text{if } r = 0 \\ 0, & \text{otherwise} \end{cases}. \tag{3.88}$$

For more details about evaluating these quantities, see Section A.2.

Since the matrix $\mathbb{M}$ is the identity matrix, then the time dependent ODEs of (3.85), can be identified with (2.10), where the matrix L and the function F are defined as follows

$$L \equiv L(\tilde{t}) = -\mathfrak{S}(\tilde{t}), \quad F = N(X, K_r).$$

Because the matrix $\mathfrak{S}$ depends on the time, $\tilde{t}$, the system of ODEs (3.85) subject to the initial condition (3.87) can be solved with the time discretization method discussed in Chapter 2, with the following approximation

$$\forall \tilde{t} \in [\tilde{t}_n, \tilde{t}_{n+1}], \qquad L \approx \mathfrak{S}(\tilde{t}_{n+1}).$$

Throughout the resolution of the system of ODEs (3.85), the dimensionless current is simultaneously computed with (3.57). These time solvers also require the computation of the Jacobian, $\partial_X N$ of the source term N at each time step.

Computation of the source term

The source term $N(X, K_r)$ can easily be formed once the components of the vector $N_1(X)$ are computed. In order to compute these components, we first rewrite (3.86) as a linear combination of the integral of the triple product of Legendre polynomials

(3.90), where the coefficients are in term of the components of X

$$\begin{aligned}
N_{1,l}^{j} &= \int_{\Omega} \tilde{C}_{\mathbf{Q}^+}\tilde{C}_B\Phi_l^j(z)dz = \int_{I_j} \tilde{C}_{\mathbf{Q}^+}\tilde{C}_B\Phi_l^j(z)dz \\
&= \int_{I_j} \left(\sum_{r=0}^{k_j} \alpha_r^{+,j}\Phi_r^j(z)\right)\left(\sum_{m=0}^{k_j} \beta_m^{-,j}\Phi_m^j(z)\right)\Phi_l^j(z)dz \\
&= \sum_{r=0}^{k_j}\sum_{m=0}^{k_j} \alpha_r^{+,j}\left(\int_{I_j}\Phi_r^j\Phi_m^j\Phi_l^j\right)\beta_m^{-,j} \\
&= \sum_{r=0}^{k_j}\sum_{m=0}^{k_j} \alpha_r^{+,j}\left(\frac{h_j\Gamma^j_{\{r,m,l\}}}{2}\int_I P_rP_mP_l\right)\beta_m^{-,j} \\
N_{1,l}^{j} &= \frac{1}{2\sqrt{h_j}}\sum_{r=0}^{k_j}\sum_{m=0}^{k_j} \alpha_r^{+,j}\left(\Gamma_{\{r,m,l\}}\int_I P_rP_mP_l\right)\beta_m^{-,j}. \qquad (3.89)
\end{aligned}$$

By introducing the $(k_j+1)\times(k_j+1)$ matrix, denoted $\mathbb{P}_l^j$, with the entries

$$\mathbb{P}_l^j(r,m) = \Gamma_{\{r,m,l\}}\int_I P_rP_mP_l, \qquad \forall r,m \in \{0,\cdots,k_j\},$$

then (3.89) can be rewritten as follows

$$N_{1,l}^{j} = \frac{1}{2\sqrt{h_j}}\alpha^{+,j^T}\mathbb{P}_l^j\beta^{-,j}\ , \forall j \in \{1,\cdots,n\}, \forall l \in \{1,\cdots,k_j\}. \qquad (3.90)$$

It is well known, see [2, 6] for more details, that the product of two Legendre polynomials can be expanded, in a series using the special Wigner $3j$ symbols, as follows

$$P_iP_m = \sum_{q=|i-m|}^{i+m}\begin{pmatrix} i & m & q \\ 0 & 0 & 0 \end{pmatrix}^2 (2q+1)P_q, \qquad (3.91)$$

where the special Wigner $3j$ symbols are computed by the following formula

$$\begin{pmatrix} i & m & q \\ 0 & 0 & 0 \end{pmatrix} = \begin{cases} \sqrt{\frac{(2g-2i)!(2g-2m)!(2g-2q)!}{(2g+1)!}}\frac{g!(-1)^g}{(g-i)!(g-m)!(g-q)!} & \text{if } J=2g \\ 0 & \text{if } J=2g+1 \end{cases},$$

for $J = i+m+q$. Therefore the integral of the triple product of Legendre polynomials

is given by

$$\int_I P_i P_m P_l = 2 \begin{pmatrix} i & m & q \\ 0 & 0 & 0 \end{pmatrix}^2, \qquad \text{if} \quad \mid m-i \mid \leq q \leq m+i.$$

Using the symmetry of the triple product and the orthogonality of the Legendre polynomials, the vector $N_1(X)$ can be efficiently computed.

Computation of the Jacobian of the source term

Since the entries of the vector, $N_1(X)$ defined by (3.90), depend only on the concentration of the species $\mathbf{Q}^+$ and B then the Jacobian of the source term $N(X, K_r)$ is given by

$$\partial_X N(X, K_r) = K_r \begin{pmatrix} 0 & 2J_1 & 2J_2 & 0 \\ 0 & -2J_1 & -2J_2 & 0 \\ 0 & -J_1 & -J_2 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \tag{3.92}$$

where the block diagonal matrices J_1 and J_2 respectively represent the Jacobian of the vector $N_1(X)$ with respect to the entries α^+ and β^- of the vector X. According to the local definition of the basis function (3.38), the block diagonal matrices J_1 and J_2 are given by

$$J_1 = \begin{pmatrix} J_1^1 & 0 & \cdots & 0 \\ 0 & J_1^2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & J_1^n \end{pmatrix}, \qquad J_2 = \begin{pmatrix} J_2^1 & 0 & \cdots & 0 \\ 0 & J_2^2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & J_2^n \end{pmatrix}, \tag{3.93}$$

with the $(k_j+1) \times (k_j+1)$ matrices J_1^j and J_2^j entries given by

$$J_1^j(m,l) = \sum_{i=0}^{k_j} \frac{\beta_i^{-,j}}{2\sqrt{h_j}} \Gamma_{\{i,m,l\}} \int_I P_i P_l P_m, \quad J_2^j(m,l) = \sum_{i=0}^{k_j} \frac{\alpha_i^{+,j}}{2\sqrt{h_j}} \Gamma_{\{i,m,l\}} \int_I P_i P_l P_m. \tag{3.94}$$

For more details about the efficient computation and update of the block matrices J_1, J_2 and the block vector N_1 throughout the numerical resolution of the model EC',

see Section A.4.

3.3.4 Numerical experiments on EC' model

We have shown in Section 3.2.4 that DG_{Impl}, compared to the `pdepe`, is the best numerical solver of the ETO model. Since the EC' model can be reduced to the ETO model, for $K_r = 0$, then we realise here some numerical experiments to compare the performance of DG_{Impl} and `pdepe` on the EC' model. By default, we set the parameters $K_0, D^+, D^-, D, K_A, K_r$ and $\tilde{C}^0_{A^{total}}$ as follow

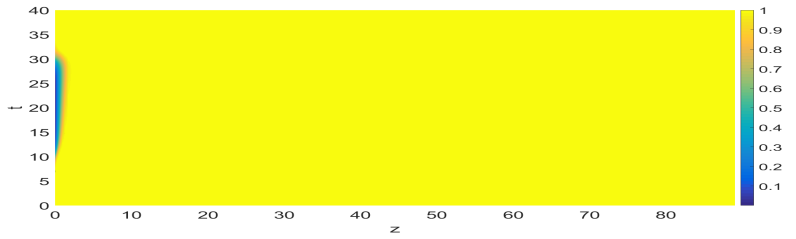
$$K_0 = 20, D^+ = 1, K_r = 10000, D^- = K_A = D = 5 \text{ and } \tilde{C}^0_{A^{total}} = 1. \tag{3.95}$$

Numerical simulation of the concentration profile of species

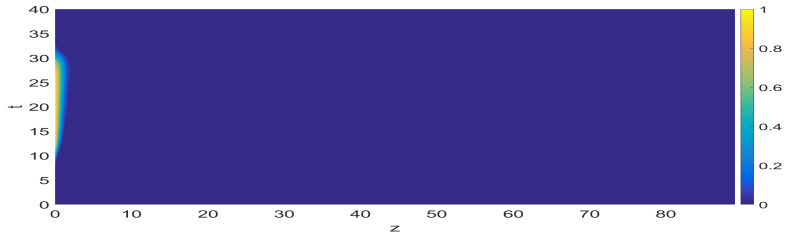
In order to validate our space discretization, we simulate the concentration profile of A, B, $\mathbf{Q}$, $\mathbf{Q}^+$ and $\mathbf{Q} + \mathbf{Q}^+$, using the DG method combined with the Implicit Euler method. If the species $\mathbf{Q}$ and $\mathbf{Q}^+$ have the same diffusion coefficient, then according to (3.79), (3.80) and the boundary conditions from (3.19) to (3.21), the sum $\tilde{S}$ of the dimensionless concentration of the species $\mathbf{Q}$ and $\mathbf{Q}^+$ is governed by (3.58), as for ETO model. We use DG_{Impl} to solve (3.85) (obtained with the DG discretization) subject to (3.87) for

$$K_0 = 20, D^+ = 1, K_r = 10000, D^- = K_A = D = 5 \text{ and } \tilde{C}^0_{A^{total}} = 1.$$

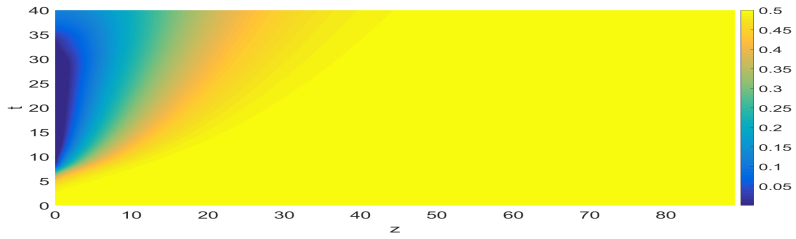
For all z and $\tilde{t}$, we respectively plot the dimensionless concentrations $\tilde{C}_{\mathbf{Q}}$, $\tilde{C}_{\mathbf{Q}^+}$, $\tilde{C}_B$, $\tilde{C}_A$ and $\tilde{S}$ in Fig 3.19(a), Fig 3.19(b), Fig 3.19(c), Fig 3.19(d) and Fig 3.19(e). Note that in Fig 3.19(e), we have $\tilde{S} = 1, \forall x, \tilde{t}$ as expected. Fig 3.5(a), Fig 3.5(b), Fig 3.19(a) and Fig 3.19(b) show that the diffusion layer of the species $\mathbf{Q}$ and $\mathbf{Q}^+$ is larger in ETO mechanism compared to EC' mechanism. This is expected since the specie $\mathbf{Q}$ is quickly refurbished by the catalytic reaction in the EC' mechanism.



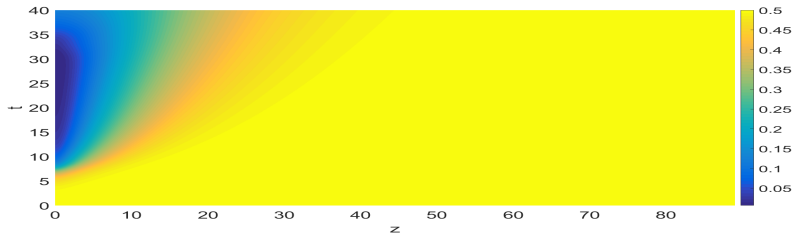
(a)



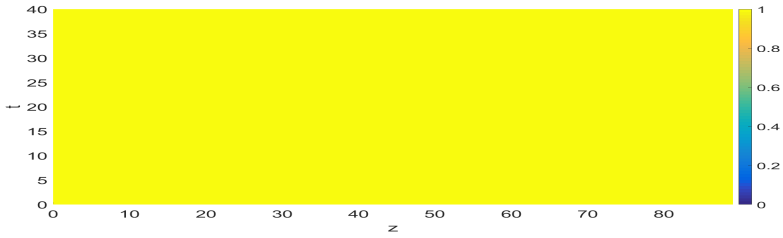
(b)



(c)



(d)



(e) Dimensionless concentration profile $\tilde{C}_{\mathbf{Q}} + \tilde{C}_{\mathbf{Q}^+}$

Figure 3.19: **Dimensionless concentration profile of the species $A, B, \mathbf{Q}$ and $\mathbf{Q}^+$ for $K_0 = 20, D^+ = 1, K_r = 10000, D^- = K_A = D = 5$ and $\tilde{C}^0_{A^{total}} = 1$.** For all z and $\tilde{t}$, we respectively plot the dimensionless concentrations $\tilde{C}_{\mathbf{Q}}$, $\tilde{C}_{\mathbf{Q}^+}$, $\tilde{C}_B$, $\tilde{C}_A$ and $\tilde{S}$ in (a), (b), (c), (d). We also plot in (e) the dimensionless concentration $\tilde{S} = \tilde{C}_{\mathbf{Q}} + \tilde{C}_{\mathbf{Q}^+}$. Note that in (e), we have $\tilde{S} = 1, \forall x, \tilde{t}$ as expected.

We compare the convergence and the efficiency of the solvers `pdepe` and DG_{Impl} with respect to the time and space discretization, while computing the concentration profile of the species $A, B, \mathbf{Q}$ and $\mathbf{Q}^+$ at the final time $\tilde{t}_2$.

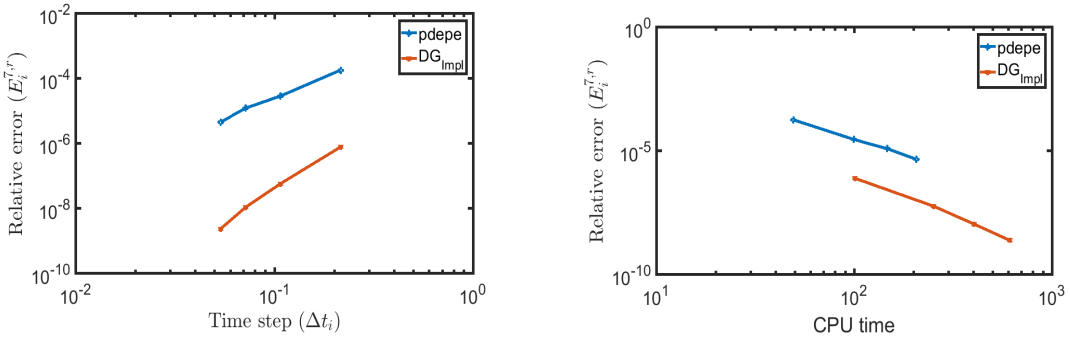
Firstly, we examine the convergence and the efficiency with respect to the time discretization. To do so, we compute the dimensionless concentration profile of the species $A, B, \mathbf{Q}$ and $\mathbf{Q}^+$ at the final time, using *pedpe* and DG_{Impl}, for the each time step $\Delta t_i = 2^i \Delta t_0$, for $i = 0, \cdots, 4$. By considering the concentration profile associated to the finest time step Δt_0 as exact concentration, we compute the relative error associated to the time step Δt_i, $\; i \in \{1, \cdots, 4\}$ by

$$E_i^{7,r} = 100 \frac{\sqrt{\sum_{j=A,B,\mathbf{Q},\mathbf{Q}^+} \left| C_j^{0,r} - C_j^{i,r} \right|^2_{L^2(\Omega)}}}{\sqrt{\sum_{j=A,B,\mathbf{Q},\mathbf{Q}^+} \left| C_j^{0,r} \right|^2_{L^2(\Omega)}}}, \tag{3.96}$$

for the each solver $r = \texttt{pdepe}, DG_{Impl}$. For the time step $\Delta t_0 = 0.0269$, we plot $E_i^{1,r}$ vs Δt_i in Fig 3.20(a) and $E_i^{1,r}$ vs CPU time in Fig 3.20(b). Note that in Fig 3.20(a), the relative error $E_i^{1,r}$ decrease with the time step and the curves associated to the both solvers are parallel. We can conclude that the solvers *pedpe*, DG_{Impl} converge in time and have the same order of convergence with respect to the time step. Note that in Fig 3.8(b), for a given relative E^0, we have

$$T^0_{\texttt{pdepe}} > T^0_{DG_{Impl}}, \tag{3.97}$$

where T_r^0 is the CPU time spends by the solver $r = \texttt{pdepe}, DG_{Impl}$ for the computation of $\tilde{C}_A, \tilde{C}_B, \tilde{C}_{\mathbf{Q}}$ and $\tilde{C}_{\mathbf{Q}^+}$ at the time $\tilde{t} = \tilde{t}_2$ such that $E_{i_0}^{1,r} = E^0$. Thus DG_{Impl} is the most efficient solver with respect to the time discretization.



(a) (b)

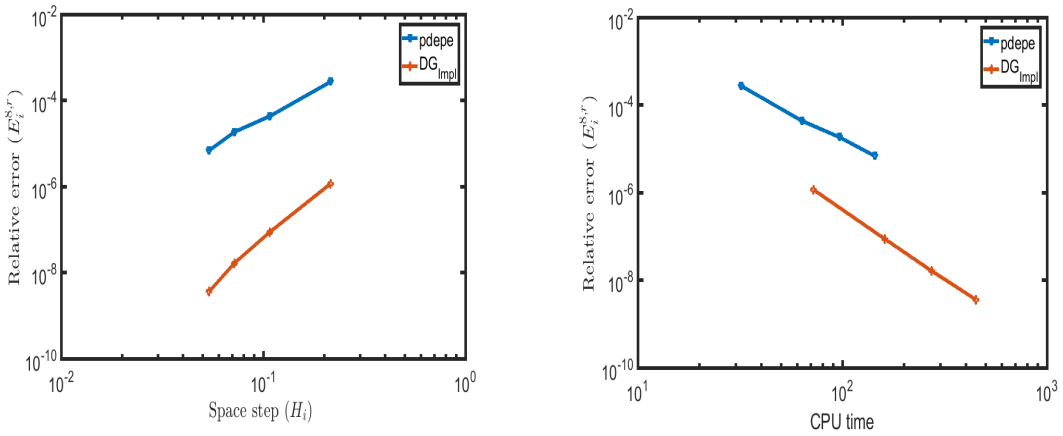
Figure 3.20: **Convergence and the efficiency with respect to the time discretization while computing $\tilde{C}_A$, $\tilde{C}_B$, $\tilde{C}_{\mathbf{Q}}$ and $\tilde{C}_{\mathbf{Q}^+}$ with the solvers pdepe and DG_{Impl}.** The convergence and the efficienciness in this case are respectively illustrated by plotting $E_i^{1,r}$ vs Δt_i in (a) and $E_i^{1,r}$ vs CPU time in (b) for $i = 1, \cdots, 4$. (a) shows that the solvers *pedpe* and DG_{Impl} have the same order of convergence with respect to the time step, but DG_{Impl} is more accurate. (b) shows that DG_{Impl} is more efficient, with respect to the time step, to compute the concentration of the species $\mathbf{Q}$ and $\mathbf{Q}^+$ at the final time $\tilde{t}_2$.

Finally, we examine the convergence and efficiency with respect to the mean of the space step by computing the concentration profile of the species $A, B, \mathbf{Q}$ and $\mathbf{Q}^+$ at the final time, using *pedpe* and DG_{Impl}, for the each partition Ω_i, for $i = 1, \cdots, 5$. The partitions Ω_i , for $i = 2, \cdots, 5$ are obtained by splitting each element of Ω_1 into i equidistant elements, for a given partition Ω_1. By considering the concentration profile associated to the finest space step H_5 of the partition Ω_5 as exact concentration, we compute the relative error associated to the mean of the space step, H_i of the partition Ω_i, $i \in \{1, \cdots, 4\}$, as follow

$$E_i^{8,r} = 100 \frac{\sqrt{\sum_{j=A,B,\mathbf{Q},\mathbf{Q}^+} \left| C_j^{5,r} - C_j^{i,r} \right|^2_{L^2(\Omega_i)}}}{\sqrt{\sum_{j=A,B,\mathbf{Q},\mathbf{Q}^+} \left| C_j^{5,r} \right|^2_{L^2(\Omega_5)}}}, \tag{3.98}$$

for the each solver $r = \texttt{pdepe}, DG_{Impl}$. For $H_5 = 0.0582$, we plot $E_i^{8,r}$ vs H_i in Fig 3.21(a) and $E_i^{1,r}$ vs CPU time in Fig 3.21(b). Note that in Fig 3.21(a), the relative error $E_i^{2,r}$ decrease with the mean of the space step H_i for all $r = \texttt{pdepe}, DG_{Impl}$. It also shows that $E_i^{2,\texttt{pdepe}} > E_i^{2,DG_{Impl}}$ for a given time step Δt. Therefore the solver DG_{Impl} is more accurate than pdepe with respect to the space discretization. Fig 3.21(b) shows that for a given relative E^0, we have $T^0_{\texttt{pdepe}} >$

$T^0_{DG_{Impl}}$, where T^0_r is the CPU time spends by the both solvers for the computation of the dimensionless concentration $\tilde{C}_A, \tilde{C}_B, \tilde{C}_{\mathbf{Q}}$ and $\tilde{C}_{\mathbf{Q}^+}$ such that $E^{8,r}_{i_0} = E^0$ at the time $\tilde{t} = \tilde{t}_2$. Thus DG_{Impl} is the most efficient solver with respect to the space discretization.



(a) (b)

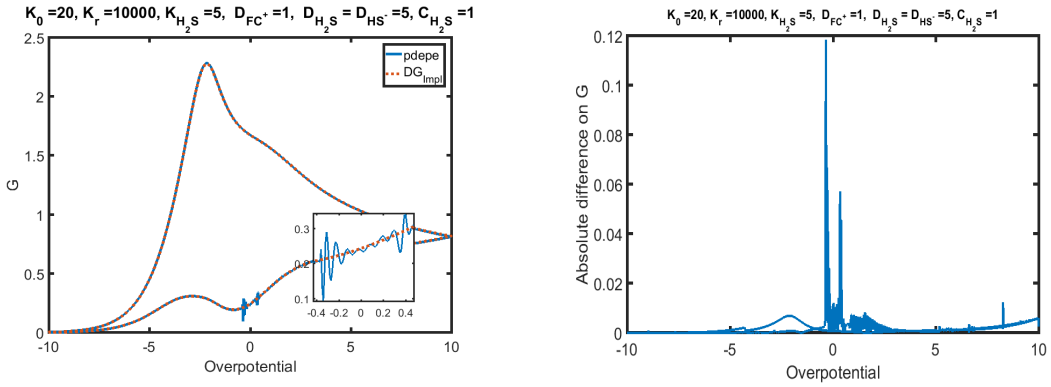
Figure 3.21: **Convergence and the efficiency with respect to the space discretization while computing $\tilde{C}_A$, $\tilde{C}_B$, $\tilde{C}_{\mathbf{Q}}$ and $\tilde{C}_{\mathbf{Q}^+}$ with the solvers `pdepe` and DG_{Impl}.** The convergence and the efficiency are respectively illustrated by plotting $E^{1,r}_i$ vs H_i in (a) and $E^{1,r}_i$ vs CPU time (b). (a) shows that the solvers *pedpe* and DG_{Impl} have the same order of convergence with respect to the space step. (b) shows that DG_{Impl} is more efficient, with respect to the space step, to compute the concentration of the species $A, B, \mathbf{Q}$ and $\mathbf{Q}^+$ at the final time $\tilde{t}_2$.

Numerical simulation of the dimensionless voltammogram

We compare the voltammograms obtained using DG_{Impl} and Matlab's solver `pdepe`. This comparison is illustrated in Fig 3.22. We plot in Fig 3.22(a) both simulated voltammograms and the zoom of a portion of their curve. Note that in Fig 3.22(a), the voltammogram obtained with `pdepe` present some oscillations. We plot in Fig 3.22(b), the absolute value of the difference between both voltammograms. It shows that despite the oscillation of the voltammogram obtained with `pdepe`, both voltammograms are almost the same since we have

$$\left| G_{\texttt{pdepe}}(t_n) - G_{DG_{Impl}}(t_n) \right| < 0.12,$$

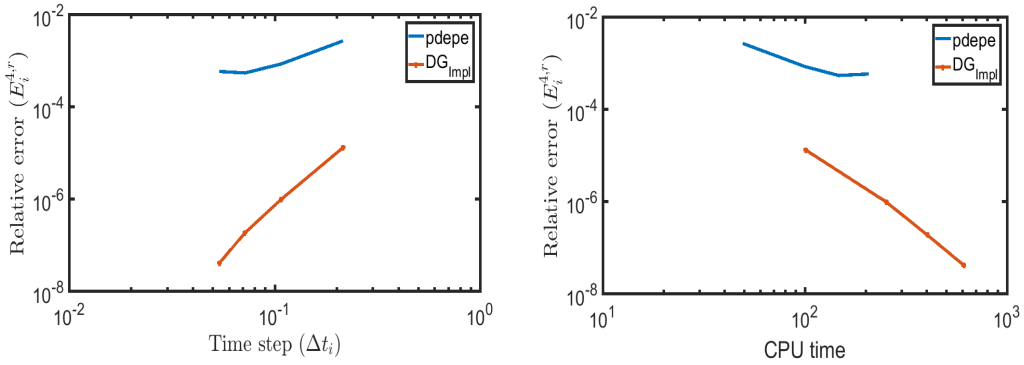
where $G_{\texttt{pdepe}}$ and $G_{DG_{Impl}}$ respectively represent the current obtained with `pdepe` and DG_{Impl}.



(a) (b)

Figure 3.22: **Comparison of the dimensionless voltammograms obtained by DG_{Impl} and Matlab's solver `pdepe` for the EC' model with $K_0 = 20, D^+ = 1, D^- = D = K_A = 5, K_r = 10000$ and $\tilde{C}^0_{A^{total}} = 1$.** We plot in (a) both voltammograms and a the highlight of their portion. See in (a) that the voltammogram obtained with `pdepe` present some oscillations but the one obtained with DG_{Impl} doesn't. We plot in (b) the absolute value of the difference between both voltammograms. it shows that despite the oscillation of the voltammogram obtained with `pdepe`, both voltammograms are almost everywhere the same.

We now examine the convergence and the efficiency of the solvers *pedpe* and DG_{Impl} with respect to the time and space discretization, while computing the dimensionless current. To do so, we firstly simulate the dimensionless voltammogram using the solvers *pedpe* and DG_{Impl} for all time step $\Delta t_i = 2^i \Delta t_0$, for $i \in \{0, \cdots, 4\}$. By considering the dimensionless current associated to the finest time step Δt_0 as the exact dimensionless current, we compute the relative error associated to the time step $\Delta t_i = 2^i \Delta t_0$, for $i \in \{1, \cdots, 4\}$ using (3.69). For $\Delta t_0 = 0.0269$, we plot $E_i^{4,r}$ vs Δt_i in Fig 3.23(a) and $E_i^{4,r}$ vs CPU time in Fig 3.23(b) for $i \in \{1, \cdots, 4\}$. Note that in Fig 3.23(a), the relative error $E_i^{4,r}$ tend to zeros with the time step for all $r =$ `pdepe`, DG_{Impl}. therefore both solvers converge with respect to the time step while simulating the voltammogram. Fig 3.23(b) shows that for a given value E_0, DG_{Impl} compared to `pdepe` will spend less time to simulate the dimensionless voltammogram with a relative error equal E_0. Thus DG_{Impl} is more efficient, with respect to the time discretization, than `pdepe` to simulate the dimensionless voltammogram.

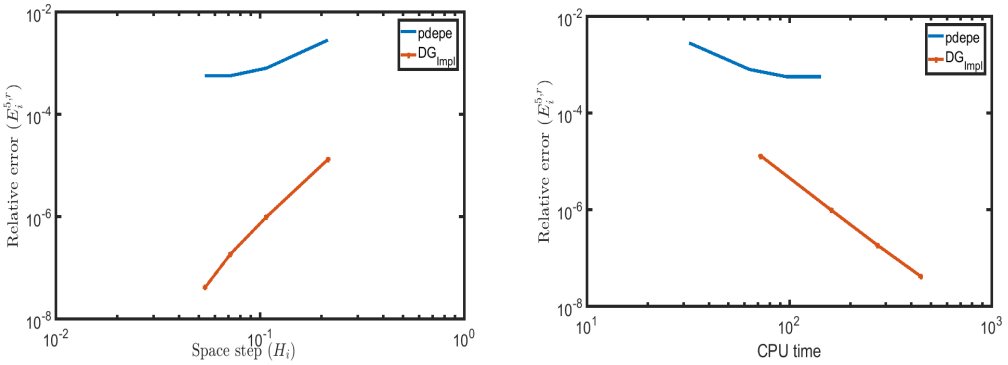


(a) (b)

Figure 3.23: **Convergence and the efficiency of `pdepe` and DG_Impl with respect to the time discretization while simulating the dimensionless voltammogram**. We plot $E_i^{4,r}$ vs Δt_i in (a) and $E_i^{4,r}$ vs CPU time in (b) for $i \in \{1, \cdots, 4\}$. Note that in (a), the relative error $E_i^{4,r}$ tend to zeros with the time step for both solvers, meaning that the solvers converge with respect to the time step while simulating the voltammogram. (b) shows that for a given value E_0, DG_{Impl} compared to `pdepe` will spend less time to simulate the dimensionless voltammogram with a relative error equal E_0. Thus DG_{Impl} is more efficient with respect to the time discretization while simulating the dimensionless voltammogram.

Finally, we examine the convergence and the efficiency of `pdepe` and DG_Impl with respect to the mean of the space steps while simulating the dimensionless voltammogram We simulate the dimensionless voltammogram using the solvers *pedpe* and DG_{Impl} for all partition Ω_i, $i = 1, \cdots, 5$. The partitions Ω_i , for $i = 2, \cdots, 5$ are obtained by splitting each element of Ω_1 into i equidistant elements, for a given partition Ω_1. By considering the dimensionless voltammogram associated to the finest mean space step H_5 of the partition Ω_5 as exact dimensionless voltammogram, we compute the relative error $E_i^{5,r}$ associated to the mean space step H_i of the partition Ω_i, $i \in \{1, \cdots, 4\}$ using (3.70). For $H_5 = 0.0582$, we plot $E_i^{5,r}$ vs H_i in Fig 3.24(a) and $E_i^{5,r}$ vs CPU time in Fig 3.24(b) for $i = 1, \cdots, 4$. Note that in Fig 3.24(a), the relative error $E_i^{5,r}$ tend to zeros with the mean space step for the solvers `pdepe` and DG_{Impl}. Then the both solvers converge with respect to the mean space step while simulating the dimensionless voltammogram. Fig 3.14(b) shows that for a given value E_0, DG_{Impl} compared to `pdepe` will spend less time to simulate the dimensionless voltammogram with a relative error equal E_0. Thus DG_{Impl} is more efficient than `pdepe` while simulating the dimensionless

voltammogram.



(a) (b)

Figure 3.24: **Convergence and the efficiency of `pdepe` and DG_{Impl} with respect to the mean of the space steps while simulating the dimensionless voltammogram**. We plot $E_i^{5,r}$ vs H_i in (a) and $E_i^{5,r}$ vs CPU time in (b) for $i = 1, \cdots, 4$. Note that in (a), the relative error $E_i^{5,r}$ tend to zeros with the mean space step for `pdepe` and DG_{Impl}. Then both solvers converge with respect to the mean space step while simulating the dimensionless voltammogram. (b) shows that for a given value E_0, DG_{Impl} compared to `pdepe` will spend less time to simulate the dimensionless voltammogram with a relative error equal E_0. Thus DG_{Impl} is more efficient with respect to the space discretization while simulating the dimensionless voltammogram.

3.4 Summary

In this chapter, we have developed step by step a novel numerical method, based on the DG space discretization, to solve the governing equations of the cyclic voltammetry models and simulate its signal response. The finite DG space is constructed with the orthonormal shifted Legendre polynomials, which reduced the mass matrix to the identity matrix.

To that end, we initially investigate in Section 3.2, the DG method for the ETO models, since it has an analytical results from Aoki et al.[9], that can be used to validate the numerical solution. We introduced the dimensionless parameters to transform the linear governing equations into the dimensionless one, which is a time dependent PDEs. Using the DG space discretization, we transformed the dimensionless governing equations into an ODEs. The obtained ODEs is then solved with implicit Euler or Exponential time differencing method. We performed various

numerical experiments in Section 3.2.4, to validate and compare our solver with Matlab's solver `pdepe`. Contrary to expectations, while combine with the DG spatial discretization, the standard implicit time integrator out performs the methods such as the ETD method and adaptive time stepping method $ode15s$. We also see that the DG method performs better than the standard FE method.

We finally investigate in Section 3.3, the DG method for the EC' models, which is described by a non linear equations. The same process as for ETO model is used to solve the dimensionless equations of the EC' model. Once again, various numerical experiments performed in Section 3.3.4, have shown that our novel numerical method, constructed by the combination of DG space discretization and Euler time discretization, is much more efficient to investigate EC' model.

Now that an efficient solver is proposed to simulate the signal response of the cyclic voltammetry for some given parameters, we will derive in the next chapter the adjoint equation of the inverse model. The adjoint equation will allow us the compute efficiently the gradient necessary to invert the forward model, while using the gradient descent algorithm.

Chapter 4

One dimension Inversion of Cyclic Voltammetry models

Contents

Central to this chapter is the construction of the numerical process to find all the parameters of the cyclic voltammetry models ETO and EC' (investigated in the previous chapter) from the results of the measurement of the current. This approach seeks the quantitative agreement of the modelling of the cyclic voltammetry with results of the experimental measurement of the current. To achieve this goal, the following steps are taken: we give in Section 4.1 an overview of inverse theory and specifically define the inversion problem of cyclic voltammetry models. Then we show how the adjoint method is used to solve our inversion problem. Finally we combine the DG method, Implicit Euler method and the adjoint method to respectively invert ETO and EC' model in Section 4.3 and Section 4.4. The results of these numerical inversions are then compared to the results obtained using the MATLAB code `pdepe` with the finite difference method (FD) used to compute the

gradient.

The novel contribution here is the inversion of the cyclic voltammetry models with the combination of the DG discretization, Impl time integrator and the adjoint method. The performance of our numerical inversion method is tested against a commercial MATLAB code for the inversion of synthetic data.

4.1 Introduction to the inverse problem

Generally, the purpose of collecting data is to gain meaningful information about a physical system or phenomenon of interest. However, in many situations the quantities that we wish to determine, which we call model parameters, are different from the quantities we are able to measure, which we call data. If the data depends, in some way, on the model parameters, then the data at least contains some information about the model parameters.

The forward problem is defined as mapping of model parameters in a functional space $\mathfrak{P}$, typically a Banach or Hilbert space, to the space of data $\mathfrak{G}$, typically another Banach or Hilbert space. It can be mathematically described as follow

$$G = \mathfrak{F}(p), \quad \text{for } p \in \mathfrak{P},\ G \in \mathfrak{G}, \tag{4.1}$$

where $\mathfrak{F}$ is a known function, p represent the model parameters and G is the measured data [144, 21].

Starting with knowledge of the measured data G in $\mathfrak{G}$, the problem of trying to reconstruct the model parameters p in $\mathfrak{P}$ is called an inverse problem such that (4.1) holds or an approximatively holds due to the error in the measurement. However, the resolution of an inverse problem is capable of doing more than estimating model parameters. It can also be used to bound the range of acceptance of model parameters, estimate the uncertainties in the model parameters, to do the sensitivity analysis of the data or to find what kind of data are best suited to determine the set of model parameters, see for example [210, 119, 7, 222, 209] for more details. If the inverse problem does not have a unique solution p in $\mathfrak{P}$, it is called an ill-posed

inverse problem [162].

Inverse problems are frequently used in a large variety of problems in sciences: curve fitting, Acoustic Tomography, X-ray Imaging, factor analysis [165], Gravitational waves [155], satellite navigation [112, 121], earthquake response signals [183].

4.1.1 Inversion problem of Cyclic Voltammetry models

The inversion problem for the cyclic voltammetry models consists of being given data (current) in order to find parameters in the model (ETO and EC'). This inversion problem can be rewritten as time dependent PDE-constrained optimization problem. By means of the discretization in space, this inversion problem is then equivalent to an ODE-constrained optimization problem defined as follows

$$\begin{aligned} &\arg\min_{p\in\mathfrak{P}} \left\{ F(x,p) = \int_0^b f(x,p,t)dt \right\}, \\ &\text{such that } \begin{cases} h(x,\dot{x},p,t) = 0 \\ g(x(t=0),p) = 0 \end{cases} \end{aligned} \tag{4.2}$$

where $x \in \mathbb{R}^{n_x}$, a vector-valued function of the time t in $[0, b]$ with time derivative denoted $\dot{x}$, is the solution of the ODE in its implicit form $h(x, \dot{x}, p, t) = 0$ subject to the initial condition $g(x(t = 0), p) = 0$ parametrized by a vector of unknown parameters $p \in \mathfrak{P} \subset \mathbb{R}^{n_p}$. The function to minimize F is called the objective function. The function f is called the misfit function and it measures the difference between the synthetic dimensionless current and the measured dimensionless current

$$f(x,p,t) = \frac{1}{2}(G(x,p,t) - G^{obs})^2. \tag{4.3}$$

In order to solve the ODE-constrained optimization problem (4.2), a gradient-based optimisation algorithm can be used [32, 176, 175]. It requires the computation of the gradient of the objective function F with respect to the model parameters p

$$d_p F(x,p) = \left(d_{p_i} F(x,p)\right)_{i=1,\cdots,n_p}, \tag{4.4}$$

which will indicate a useful search direction. Two methods are commonly used to approximate the gradient d_pF. The first method is to use one of the finite difference formulas defined as follows for $\epsilon > 0$

$$d_{p_i}F(x,p) = \frac{F(x,p) - F(x,p-\epsilon e_i)}{\epsilon}, \tag{4.5}$$

$$d_{p_i}F(x,p) = \frac{F(x,p+\epsilon e_i) - F(x,p)}{\epsilon}, \tag{4.6}$$

$$d_{p_i}F(x,p) = \frac{F(x,p+\epsilon e_i) - F(x,p-\epsilon e_i)}{2\epsilon}, \tag{4.7}$$

where $(e_i)_{i=1,\cdots,n_p}$ represent the standard basis for Euclidean space $\mathbb{R}^{n_p}$. The finite difference formulas (4.5), (4.6) and (4.7) are respectively called backward, forward and centred finite difference formula. Therefore the computation of $d_pF(x,p)$ will require the integration of n_p additional ODEs when we use backward or forward finite difference formula and the integration of 2^{n_p} additional ODEs when we use centred finite difference formula. For large scale optimisation problems, the number n_p of parameters can be very large, and calculating the gradient using a finite difference formula becomes very expensive. The second method, called adjoint method, gives an efficient way to evaluate the gradient $d_pF(x,p)$, with a cost that is independent of the number n_p of parameters p.

4.2 Adjoint method for ODE-constrained optimization

The adjoint method for ODE-constrained optimization problem is based on developing a second ODE, called the adjoint equation, involving the Lagrangian multiplier vector, that is instrumental in the computation of the gradient $d_pF(x,p)$, see for example [49, 213] for more details.

4.2.1 Derivation of the adjoint equation

Let us introduce the Lagrangian corresponding to the optimization problem (4.2)

$$\mathcal{L} = \Gamma^T g(x(t=0), p) + \int_0^b \left[f(x,p,t) + \lambda^T h(x, \dot{x}, p, t) \right] dt \tag{4.8}$$

The vector of Lagrangian multipliers λ is a function of the time t and Γ is another vector of multipliers associated with the initial conditions, therefore not a function of time. Because the constraints $h = 0$ and $g = 0$ are satisfied by construction, we are free to set the values of λ and Γ, and we will have $F = \mathcal{L}$. Therefore the gradient $d_p F$ of the objective function F is equal to the gradient $d_p \mathcal{L}$ of the Lagrangian function $\mathcal{L}$. Taking the total derivative of (4.8), we obtain

$$d_p\mathcal{L} = \Gamma^T(\partial_{x(0)} g d_p x(0) + \partial_p g) + \int_0^b \left[\partial_x f d_p x + \partial_p f + \lambda^T(\partial_x h d_p x + \partial_{\dot{x}} h d_p \dot{x} + \partial_p h) \right] dt \tag{4.9}$$

The integrand in (4.9) contains terms in $d_p x$ and $d_p \dot{x}$. We use integration by parts to get rid of the term $d_p \dot{x}$ in the expression of $d_p \mathcal{L}$, we have

$$\int_0^b \lambda^T \partial_{\dot{x}} h d_p \dot{x} \, dt = \left[\lambda^T \partial_{\dot{x}} h d_p x \right]_0^b - \int_0^b \left[\dot{\lambda}^T \partial_{\dot{x}} h + \lambda^T d_t \partial_{\dot{x}} h \right] d_p x \, dt, \tag{4.10}$$

then substituting this result into (4.9) and collecting terms in $d_p x$, we obtain

$$\begin{aligned} d_p\mathcal{L} = \int_0^b \left[(\partial_x f + \lambda^T(\partial_x h - d_t \partial_{\dot{x}} h) - \dot{\lambda}^T \partial_{\dot{x}} h) d_p x + \partial_p f + \lambda^T \partial_p h \right] dt \\ + \lambda^T \partial_{\dot{x}} h d_p x \Big|_b + (\Gamma^T \partial_{x(0)} g - \lambda^T \partial_{\dot{x}} h) \Big|_0 d_p x(0) + \Gamma^T \partial_p g. \end{aligned} \tag{4.11}$$

Computing $d_p x$ is difficult in most cases. So to easily compute $d_p \mathcal{L}$ from (4.11), we set the coefficient of $d_p x, d_p x|_b$ and $d_p x|_0$ in (4.11) to zero, since the Lagrangian multipliers are arbitrary. Therefore we obtain the following equations

$$\Gamma^T = \lambda^T \partial_{\dot{x}} h [\partial_{x(0)} g]^{-1} \Big|_{t=0}, \quad \partial_x f + \lambda^T(\partial_x h - d_t \partial_{\dot{x}} h) - \dot{\lambda}^T \partial_{\dot{x}} h = 0, \quad \lambda(b) = 0,$$

and the gradient of the Lagrangian is then reduced to

$$d_p\mathcal{L} = \int_0^b \left[\partial_p f + \lambda^T \partial_p h\right]\ dt + \lambda^T \partial_{\dot{x}} h[\partial_{x(0)} g]^{-1} \partial_p g\Big|_{t=0} = d_p F. \tag{4.12}$$

4.2.2 Algorithm to compute the gradient d_pF

For a given parameter p, the gradient d_pF of the objective function F can be computed with the following step

1. Solve the forward problem by integrating the ODE constraint

$$h(x, \dot{x}, p, t) = 0, \quad g(x(t=0), p) = 0, \tag{4.13}$$

   from $t = 0$ to b and store the solution x_n for each time step t_n.

2. Solve the adjoint problem by integrating the adjoint equation

$$\partial_x f + \lambda^T(\partial_x h - d_t \partial_{\dot{x}} h) - \dot{\lambda}^T \partial_{\dot{x}} h = 0, \quad \lambda(b) = 0, \tag{4.14}$$

   for λ from $t = b$ to 0.

3. Compute the gradient d_pF with the following formula

$$d_p F = \int_0^b \left[\partial_p f + \lambda^T \partial_p h\right]\ dt + \lambda^T [\partial_{x(0)} g]^{-1} \partial_p g\Big|_{t=0}. \tag{4.15}$$

To improve the efficiency of our code for the numerical simulation, the integrand of the gradient d_pF is computed simultaneously at each time step t_n while integrating the adjoint equation for λ from $t = b$ to 0.

4.2.3 Relationship between forward and adjoint problem

The relationship between the forward (4.13) and the adjoint (4.14), depends only the type of ODE constraint (linear or non linear) in (4.2). So we show here how to define the adjoint equation from the explicit form of the ODE constraint in (4.2).

First case: linear first order explicit ODE constraint

In this case the function h takes the form

$$h(x, \dot{x}, p, t) = \dot{x} - \mathcal{A}(p)x - \mathcal{B}(p),$$

where the vector $\mathcal{B}$ describes the boundary conditions and the matrix $\mathcal{A}$ depends on the model parameters p. So we have $\partial_x h = -\mathcal{A}(p),\ \partial_{\dot{x}} h = I$, where I is the identity matrix. Then, by taking the transpose of (4.14) and introducing the change of variable $\bar{t} = b - t$, we obtain the adjoint problem :

$$\text{Find } \lambda \text{ such that } \begin{cases} d_{\bar{t}}\lambda = \mathcal{A}(p)^T\lambda - \partial_x f^T, \ \ \bar{t} \in [0, b] \\ \lambda(\bar{t} = 0) = 0. \end{cases} \tag{4.16}$$

The gradient $d_p F$ of the objective function is then computed by Algorithm 1, which requires the computation of the Jacobian $\partial_{x(0)} g$ and the gradients $\partial_p f, \partial_p h$ and $\partial_p g$ of the functions f, h and g.

Algorithm 1: Computation of the gradient $d_p F$ for linear constraint.

1 Solve the forward problem

$$\text{Find } x \text{ such that } \begin{cases} d_t x = \mathcal{A}x + \mathcal{B}, \ \ t \in [0, b] \\ g(x(t = 0), p) = 0. \end{cases} \tag{4.17}$$

2 Solve the adjoint problem

$$\text{Find } \lambda \text{ such that } \begin{cases} d_{\bar{t}}\lambda = \mathcal{A}^T\lambda - \partial_x f^T, \ \ \bar{t} \in [0, b] \\ \lambda(\bar{t} = 0) = 0. \end{cases} \tag{4.18}$$

3 Compute the gradient $d_p F$ of the objective function F with (4.15).

Second case: non linear first order explicit ODE constraint

In this case the function h takes the form

$$h(x, \dot{x}, p, t) = \dot{x} - \mathcal{A}x - \mathfrak{N}(x, p) - \mathcal{B}(p),$$

where $\mathfrak{N}$ is a non linear function of x which also depends on the model parameters p. So we have $\partial_x h = -\mathcal{A} - \partial_x \mathfrak{N}$ and $\partial_{\dot{x}} h = I$. Then by taking the transpose of (4.14) and introducing the change of variable $\bar{t} = b - t$, we obtain the adjoint problem :

$$\text{Find } \lambda \text{ such that } \begin{cases} d_{\bar{t}}\lambda = (\mathcal{A} + \partial_x \mathfrak{N})^T \lambda - \partial_x f^T, \quad \bar{t} \in [0, b] \\ \lambda(\bar{t} = 0) = 0. \end{cases} \tag{4.19}$$

The gradient $d_p F$ of the objective function is then computed by Algorithm 2.

Algorithm 2: Computation of the gradient $d_p F$ for non linear constraint.

1 Solve the Forward problem

$$\text{Find } x \text{ such that } \begin{cases} d_t x = \mathcal{A}x + \mathfrak{N}(x, p) + \mathcal{B}, \quad t \in [0, b] \\ g(x(t = 0), p) = 0. \end{cases} \tag{4.20}$$

2 Solve the adjoint Problem

$$\text{Find } \lambda \text{ such that } \begin{cases} d_{\bar{t}}\lambda = (\mathcal{A} + \partial_x \mathfrak{N})^T \lambda - \partial_x f^T, \quad \bar{t} \in [0, b] \\ \lambda(\bar{t} = 0) = 0. \end{cases} \tag{4.21}$$

3 Compute the gradient $d_p F$ of the objective function F with (4.15).

4.3 Numerical inversion of the ETO model

We have shown, in Section 3.2, that we can generate the synthetic response (current) of the ETO model for a given value of the parameters (4.22).

$$p^{ETO} = (\nu, C_m, T, P_1, P_2, K_0, D^+), \tag{4.22}$$

where ν is the scan rate, C_m is the initial concentration, T is the temperature, K_0 is dimensionless electron transfer rate, D^+ is dimensionless diffusion coefficient of the specie $\mathbf{Q}^+$, P_1 and P_2 are respectively dimensionless initial and reverse potential.

For the experimental data collected for the ETO model, we assume that the parameters $p^{known} = (\nu, C_m, T, P_1, P_2)$ are known. Therefore the inverse problem of ETO model is given by (4.2), with the model parameter $p = (K_0, D^+)^T$, the final time $b = 2(P_2 - P_1)$, the function h and g, respectively derived from (3.53) and

(3.54), take the following form

$$h(x, \dot{x}, p, t) = \dot{x} + (\mathcal{L}_0 + \mathcal{L}_1(t))x, \quad g(x\Big|_{t=0}, p) = x\Big|_{t=0} - x^0,$$

where x^0 is a vector depending on p and $x = (\alpha, \alpha^+)^T$ is a coupled component α and α^+ respectively of the species $\mathbf{Q}$ and $\mathbf{Q}^+$ in the DG finite space. Since the function h is linear, then the gradient d_pF of the objective function F is computed by Algorithm 1 with the matrix $\mathcal{A}$ and the vector $\mathcal{B}$ defined as follows

$$\mathcal{A} = -(\mathcal{L}_0 + \mathcal{L}_1(t)); \quad \mathcal{B} = 0. \tag{4.23}$$

The Algorithm 1 requires the computation of the Jacobian $\partial_{x(0)}g$, the gradient $\partial_p f, \partial_p h$ and $\partial_p g$ respectively of the function f, h and g.

4.3.1 Computation needed to invert ETO model

To numerically invert the ETO model, we compute here the explicit form of the additional entities needed. The Jacobian $\partial_x f(x, p, t)$ is given by

$$\partial_x f(x, p, t) = (G(x, p, t) - G^{obs})(K_f(t)Y_1, -K_b(t)Y_1),$$

where Y_1 is a $1 \times n_T$ vector defined as follows

$$Y_1(r) = \begin{cases} (-1)^r \Gamma^1_{\{r\}} & \text{if } r \in 0, \cdots, k_1 \\ 0 & \text{otherwise} \end{cases}. \tag{4.24}$$

The gradient $\partial_p f(x, p, t) = (\partial_{K_0} f(x, p, t), \partial_{D^+} f(x, p, t))$ is given by

$$\partial_{K_0} f(x, p, t) = \frac{G(x, p, t)}{K_0}(G(x, p, t) - G^{obs}), \quad \partial_{D^+} f(x, p, t) = 0.$$

From (3.23) and (3.52), we can conclude that only the matrices $\mathcal{L}_1(t)$ and $\mathcal{L}_0$ depend respectively on the positive parameter K_0 and D^+. Then the partial derivative

of h with respect to the parameters K_0 and D^+ are respectively defined as follows

$$\partial_{K_0} h(x, \dot{x}, p, t) = -\partial_{K_0} \mathcal{L}_1(t) x = \frac{G(x, p, t)}{K_0} (Y_2, -Y_2)^T,$$
$$\partial_{D^+} h(x, \dot{x}, p, t) = -\partial_{D^+} \mathcal{L}_0 x = (0, L_0^{s,\sigma_0} \alpha^+)^T,$$

where the entries of the vector $Y_2 \in \mathbb{R}^{1 \times n_T}$ are defined by

$$Y_2(r) = \begin{cases} (-1)^r & \text{if } r \in 0, \cdots, k_1 \\ 0 & \text{otherwise} \end{cases}. \tag{4.25}$$

Since the function g depends only on $x(0)$ for the ETO model, then the Jacobian $\partial_{x(0)} g$ and the gradient $\partial_p g$ are given by

$$\partial_{x(0)} g = \mathcal{I}, \qquad \partial_p g \Big|_{t=0} = (0, 0), \tag{4.26}$$

and according to (4.26), the gradient $d_p F$, given in (4.15), can be reduced to

$$d_p F = \int_0^{2t_\lambda} \left[\partial_p f + \lambda^T \partial_p h \right] \, dt.$$

Once the adjoint equation associated to the ETO model is solved for the Lagrangian coefficient λ, then the entries of the gradient $d_p F = (d_{K_0} F, d_{D^+} F)$ of the objective function are given by

$$d_{K_0} F(x, p) = \int_0^{2t_\lambda} \underbrace{\frac{G(x, p, t)}{K_0} \left[(G(x, p, t) - G^{obs}) + (\lambda_1 - \lambda_2) Y_2 \right]}_{f_1(x,p,t)} \, dt$$
$$d_{D^+} F(x, p) = \int_0^{2t_\lambda} \underbrace{\lambda_2 L_0^{s,\sigma_0} \alpha^+}_{f_2(x,p,t)} \, dt,$$

where $\lambda^T = (\lambda_1, \lambda_2)$ with $\lambda_i \in \mathbb{R}^{1 \times n_T}$ for all $i = 1, 2$.

4.3.2 Numerical experiment on the inversion of the synthetic response of ETO model

The purpose in this section is to first validate the adjoint method for the computation of the gradient of the objective function in the case of the ETO model. Finally, we combine the DG method, Impl method and the adjoint method to invert the synthetic data from ETO model, considered as observed data. Unless stated, we assume that the model parameters $p = (K_0, D^+)$ are bounded as follows

$$K_0 \in [0.1, 50],\ D^+ \in [0.1, 30], \tag{4.27}$$

and the synthetic dimensionless current, considered as the observed data G^{obs}, is obtained by using the dimensionless parameters $K_0^{obs} = 8,\ D_{obs}^+ = 10$ for the ETO model. In order to make the optimization method more efficient, we introduce the normalised parameters $\hat{p}_i = (p_i - a_i)(b_i - a_i)^{-1}, i \in \{1, 2\}$, for all parameter $p_i \in [a_i, b_i], i \in \{1, 2\}$. Thus, we have $\hat{p}_i \in [0, 1]$ for all $i = 1, 2$ and the entries of the gradient $d_{\hat{p}}F$ of the objective function F are $d_{\hat{p}_i}F = (b_i - a_i)d_{p_i}F, i \in \{1, 2\}$. Let us respectively denote $\hat{K}_0^{obs}$ and $\hat{D}_{obs}^+$ the normalized values of the default parameters K_0^{obs} and D_{obs}^+.

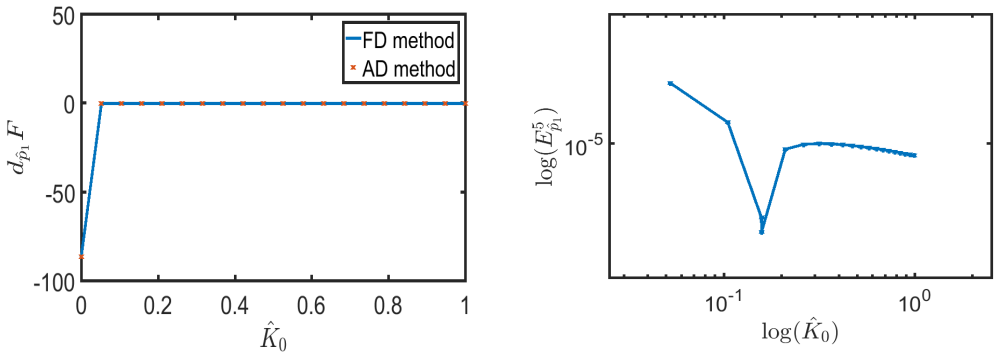
Numerical experiment 1: Test of the adjoint method for ETO model

To validate the adjoint method for the ETO model, we compare the entries of gradient $d_{\hat{p}}F$ of the objective function F obtained by using Algorithm 1 and (4.7) i.e. the centred the finite difference formula with $\epsilon = 10^{-4}$. We focus here on the first entry, denoted $d_{\hat{p}_1}^n F$, of the gradient $d_{\hat{p}}F$ at all the point $\hat{p}^n = (\hat{K}_0^n = n \times (b_1 - a_1)^{-1}, \hat{D}_{obs}^+)$, for all $n \in \{0, \cdots, 20\}$. Then, we compute the difference

$$E_{\hat{p}_1^n}^5 = \left| d_{\hat{p}_1}^{n,1} F - d_{\hat{p}_1}^{n,2} F \right|,\ \forall n \in \{0, \cdots, 20\}, \tag{4.28}$$

where $d_{\hat{p}_1}^{n,1}F$ and $d_{\hat{p}_1}^{n,2}F$ are respectively the first entry of the gradient $d_{\hat{p}}F$ computed with Algorithm 1 and (4.7). We plot in Fig 4.1(a) $d_{\hat{p}_1}^{n,r}F$ against the normalized parameter $\hat{K}_0^n$ for all $r = 1, 2$. We plot in Fig 4.1(b) the difference $\log(E_{\hat{p}_1^n}^5)$ as a

function of $\log(\hat{K}_0^n)$. We can conclude from Fig 4.1(b) that $d_{\hat{p}_1}^{n,1} \approx d_{\hat{p}_1}^{n,2}$. The same procedure is followed in Section A.5.1 to compare the entry $d_{\hat{p}_2}F$ of the gradient $d_{\hat{p}}F$ using the adjoint method or the centred finite element method (the result of this comparison is shown in Fig A.1).



(a) (b)

Figure 4.1: **Comparison of the gradient using adjoint and finite difference method for the ETO model.** In (a) we plot the total derivative $d_{\hat{p}_1}F$, compute with the adjoint and finite difference method, as a function of the parameter $\hat{K}_0$. In (b) we plot the logarithm of the difference $E^5_{\hat{p}_1^n}$ as a function of the logarithm of $\hat{K}_0^n$.

According to [185], by a Taylor expansion, for a given parameter p and the unit vector $\mathcal{D}$ there exist a constant a_p and b_0 such that for all $\alpha \in [0, b_0]$

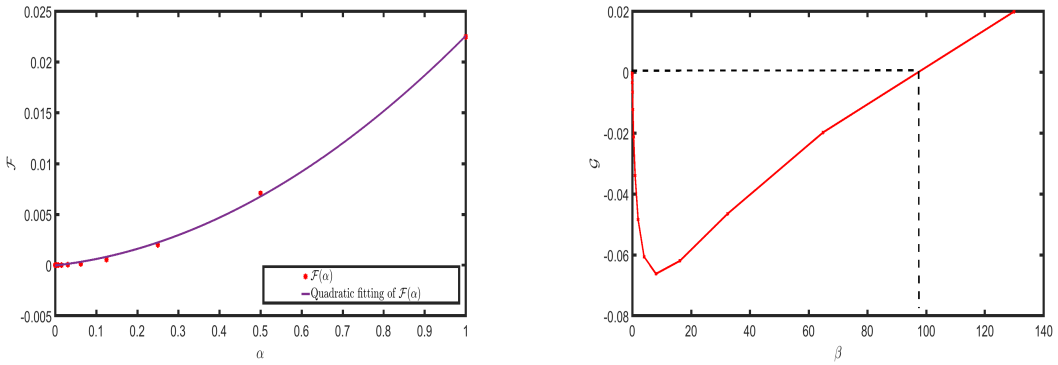
$$\mathcal{F}(\alpha) = \Big| F(p + \alpha\mathcal{D}) - F(p) - \alpha\mathcal{D}^T d_p F(p) \Big| \approx a_p \alpha^2. \tag{4.29}$$

And if the direction $\mathcal{D}$ is equal to the opposite direction of $d_pF(p)$, then by the gradient descent property in [32], there exist a positive constant b_1 such that for all value of β in the interval $[0, b_1]$, we have

$$\mathcal{G}(\beta) = F(p + \beta\mathcal{D}) - F(p) < 0. \tag{4.30}$$

So another approach to validate the adjoint method and its implementation for the ETO model, is to check if (4.29) and (4.30) hold using the adjoint method. To this end, we set $p = (25, 1)$ and compute $d_pF(p)$ with Algorithm 1. By considering $\mathcal{D} = -d_pF(p) \parallel d_pF(p) \parallel^{-1}$, we compute $\mathcal{F}(\alpha^n = 2^{-n})$ and $\mathcal{G}(\beta^n = 100 \times 2^{-n})$ for

all $n \in \{0, \cdots, 32\}$. In Fig 4.2(a), we plot $\mathcal{F}(\alpha^n)$ against α^n and fit the result with MATLAB quadratic function. Note from Fig 4.2(a) shows that (4.29) holds for $b_0 = 1$. We also plot in Fig 4.2(b) $\mathcal{F}(\beta^n)$ against β^n, which shows that (4.30) holds for $b_1 = 80$.



(a) (b)

Figure 4.2: **Test of Taylor expansion and gradient descent property using the adjoint method for ETO model.** In (a), we plot $\mathcal{F}(\alpha^n)$ against α^n and fit the result with MATLAB quadratic function. It shows that Taylor expansion holds for $b_0 = 1$. In (b), we plot $\mathcal{G}(\beta^n)$ against β^n, which shows that the gradient descent property holds for $b_1 = 80$.

Numerical experiment 2: Inversion of the ETO synthetic response using the DG, Impl and adjoint method.

Now that the adjoint method for ETO model and its implementation have been validated, we focus here in the inversion of the synthetic data using the DG, Impl and adjoint methods, denoted DG +AD. To do so, we first investigate the inversion of the default observed data, G^{obs}. Starting from a guess parameters $K_0 = 5.8307$ and $D^+ = 13.8814$, we solve the minimization problem using gradient descent algorithm, described in [32]. We stop the numerical inversion at the parameter p_0 when the norm of the gradient $d_pF(p_0)$ is less or equal to 5×10^{-6}. At this stage the value of the parameters are $K_0 = 7.9294$ and $D^+ = 10.0007$. We plot in Fig 4.3(a) the observed and the initial voltammograms. We plot the absolute value of the difference between the initial and observed voltammograms in Fig 4.3(b). We plot in Fig 4.3(c) the observed and the predicted voltammograms. We plot the absolute value of the difference between the observed and predicted voltammograms in Fig 4.3(d).

Then we plot in Fig 4.3(e) the path followed by the predicted parameters until the stopping criteria is satisfied. The relative error on the dimensionless current G is $E_G \approx 0.034\%$, on the dimensionless electron constant rate K_0 is $E_{K_0} \approx 0.88\%$ and on the dimensionless diffusion coefficient D^+ of the specie $\mathbf{Q}^+$ is $E_{D^+} \approx 0.0067\%$.

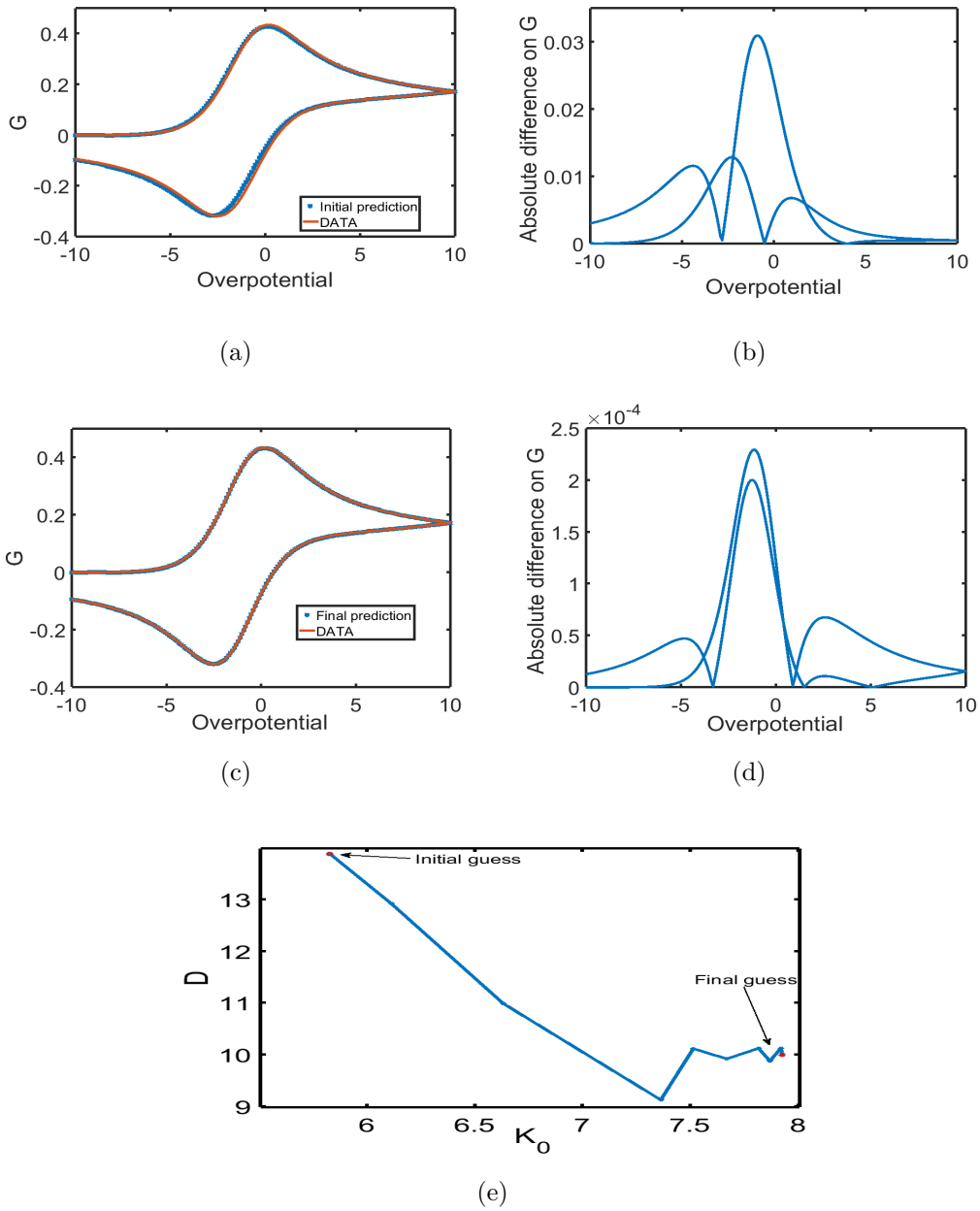


Figure 4.3: **Example of the ETO synthetic response inversion using the DG, Impl and adjoint method.** Starting from a guess parameters $K_0 = 5.8307$ and $D^+ = 13.8814$, we solve the minimization problem using gradient descent algorithm. We plot in (a) the observed and the initial voltammograms. We plot the absolute value of the difference between the initial and observed voltammograms in (b). We plot in (c) the observed and the predicted voltammograms. We plot the absolute value of the difference between the observed and predicted voltammograms in (d). then we plot in (e) the path followed by the predicted parameters from the initial guess until the stopping criteria is satisfied.

Numerical experiment 3: Efficiency of the combination of DG, Impl and adjoint method for the inversion of the synthetic response of ETO.

The goal, in this section, is to investigate the efficiency of the DG + AD method to invert the synthetic response of ETO. To that end, we compare the performance of the DG + AD against another approach, denoted FE + FD, to invert the synthetic response of ETO. The FE + FD method is the combination of the centred finite difference (FD) method for the computation of the gradient of the objective function and MATLAB code *pdepe* for the resolution of PDEs.

We then use the two approaches, i.e. DG + AD and FE + FD, to invert the synthetic responses $G_i^{obs,ETO}$ of the ETO model, associated to the model parameters $p^i = (K_0^i, D_i^+)$ generated randomly such that (4.27) holds for all i in $\{1, 2, 3, 4\}$. The results of the inversion is illustrated Tab 4.1 where p_i^{guess} is the initial guess model parameter of the descent gradient algorithm, p_i^r is the predicted model parameter obtained using the approach $r \in \{DG + AD, FE + FD\}$ for all i in $\{1, 2, 3, 4\}$.

The results are also schematically presented in Fig 4.4. We plot in Fig 4.4(a), (d), (g) and (j) the observed voltammogram and the predicted voltammograms obtained using the approaches DG + AD and FE + FD respectively for the inversion experiment $i = 1, 2, 3, 4$. Throughout each inversion, we record the CPU time. We respectively plot in Fig 4.4(b), (e), (h) and (k) the CPU time of the approaches DG + AD and FE + FD during the inversion of the observed data $G_i^{obs,ETO}$, $i = 1, 2, 3, 4$. As expected, Fig 4.4(b), (e), (h) and (k) show that **DG + AD is faster compared to FE + FD to invert the synthetic response of the ETO model**. Once the inversion of the observed data $G_i^{obs,ETO}$ is finished i.e. $p_i^r = (K_0^{r,i}, D_{r,i}^+)$ is predicted, we compute the absolute error on the dimensionless model parameters K_0, D^+

$$E_{K_0,i}^r = 100\times\left|K_0^i - K_0^{r,i}\right|\times\left|K_0^i\right|^{-1},\ E_{D^+,i}^r = 100\times\left|D_i^+ - D_{r,i}^+\right|\times\left|D_i^+\right|^{-1}, \tag{4.31}$$

and the absolute error on the dimensionless current

$$E_{G,i}^r = 100\times\left|G_i^{obs,ETO} - G_i^r\right|_{L^2(P)}\times\left|G_i^{obs,ETO}\right|_{L^2(P)}^{-1}, \tag{4.32}$$

where G_i^r is the dimensionless current obtained from the predicted model parameter p_i^r for all $r \in \{DG+AD, FE+FD\}$ and $i \in \{1,2,3,4\}$. We then plot these absolute errors for each inversion method in Fig 4.4(c), Fig 4.4(f), Fig 4.4(i), Fig 4.4(l) respectively for the inversion experiment $i = 1,2,3,4$. The scales in Fig 4.4(c), Fig 4.4(f), Fig 4.4(i), Fig 4.4(l) which show that **DG + AD, compared to FE + FD, is more accurate to invert synthetic response of the ETO model**.

		K_0	D^+
$G_1^{obs,ETO}$	p_1^{guess}	17.316915142871846	8.074766022789072
	p_1^{DG+AD}	39.130680044804464	15.674890801539632
	p_1^{FE+FD}	17.435122056473480	15.983942987120692
	p_1	39.545563811966765	15.673817496388676
$G_2^{obs,ETO}$	p_2^{guess}	15.003689359617777	10.404183098085726
	p_2^{DG+AD}	10.096972370428409	7.791932113541441
	p_2^{FE+FD}	15.797993516577412	7.773642605030885
	p_2	10.114520194105989	7.791766414718039
$G_3^{obs,ETO}$	p_3^{guess}	49.943052408344080	2.986529885053503
	p_3^{DG+AD}	48.953681873673160	11.877685496534683
	p_3^{FE+FD}	36.389928127578340	9.159860622625756
	p_3	48.988630356887040	11.877524221053571
$G_4^{obs,ETO}$	p_4^{guess}	10.681195759200980	26.997401304716050
	p_4^{DG+AD}	4.517347177066070	4.047655914506707
	p_4^{FE+FD}	10.652185573874020	26.910310766612680
	p_4	4.517328151870212	4.047665369840588

Table 4.1: **Inversion of the ETO synthetic response.** This table presents the model parameters, p_i, the initial guess, p_i^{guess}, the estimated parameters p_i^{DG+AD} and p_i^{FE+FD} respectively obtained with the approaches DG+AD and FE+FD, for the inversion of $G_i^{obs,ETO}$, $i = 1, \cdots, 4$. It shows that DG + AD is more accurate compared to FE + FD to invert synthetic response of the ETO model.

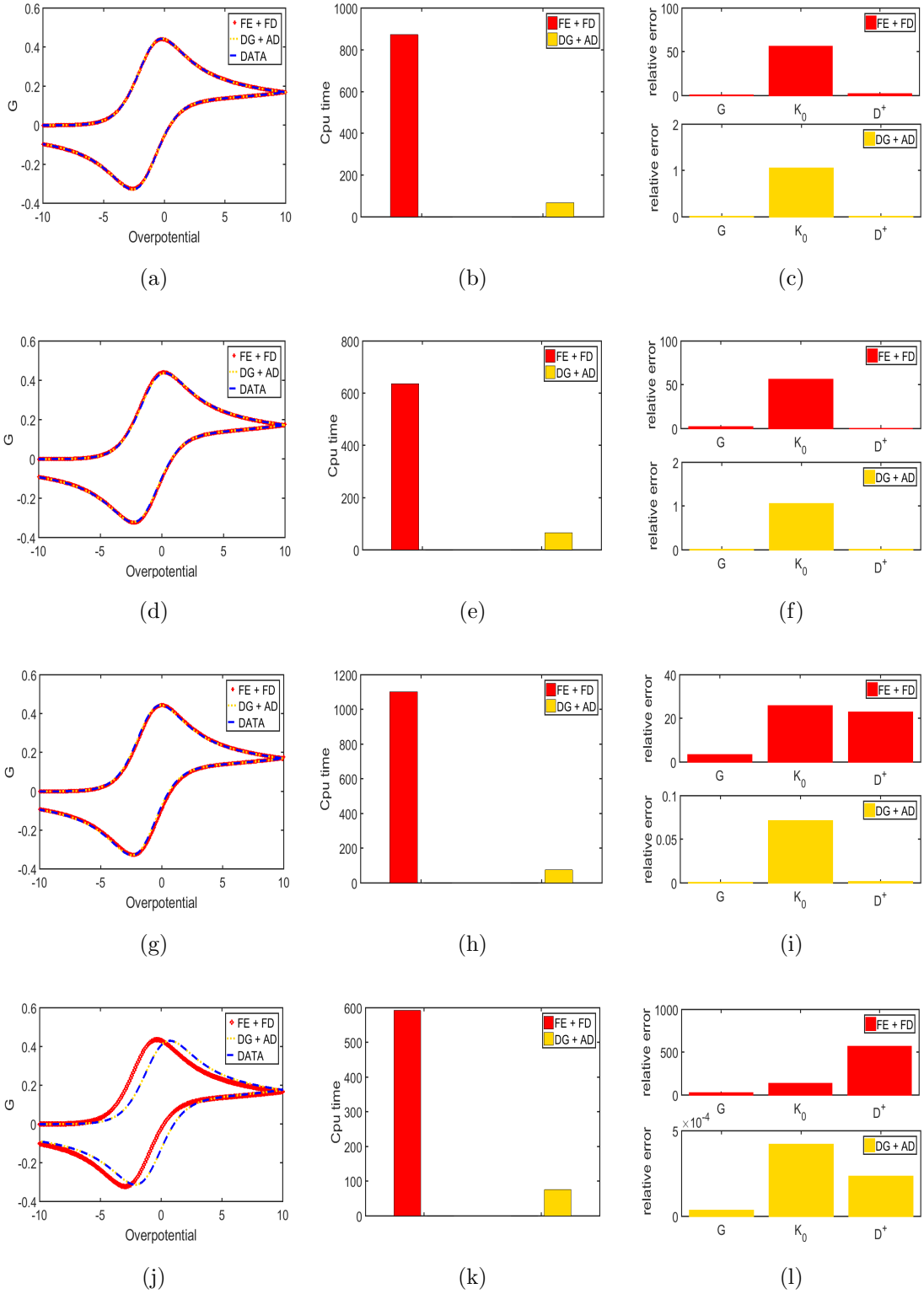


Figure 4.4: **Inversion of the ETO synthetic response.** We plot in (a), (d), (g), (j) the observed voltammogram and the predicted voltammograms obtained using DG + AD and FE + FD respectively for the inversion experiment $i = 1, 2, 3, 4$. We respectively plot in (b), (e), (h), (k) the CPU time of DG + AD and FE + FD during the inversion of the observed data $G_i^{obs,ETO}$, $i = 1, 2, 3, 4$. As expected, note from (b), (e), (h), (k) that DG + AD is faster than FE + FD. We plot in (c), (f), (i), (l) the absolute error on the current and the components of the model parameter in each experiment of inversion $i = 1, 2, 3, 4$. Note from the different scales in (c), (f), (i), (l) that DG + AD is more accurate than FE + FD.

Numerical experiment 4: Efficiency of the combination of DG, Impl and adjoint method for the inversion of the synthetic response of ETO with some additional random noise.

The purpose, in this section, is to investigate the efficiency of the DG + AD method to invert the synthetic response of ETO with some additional random noise. The motivation behind this investigation is that the experimental data usually contain some additional noise. We assume that this noise follows the Gaussian distribution. Therefore we compare the inversion of the noisy synthetic responses, $G_{noisy,i}^{obs,ETO}$, $i \in \{1, \cdots, 4\}$ using DG + AD and FE + FD approaches. The noisy synthetic responses considered here are defined from the previous inverted synthetic responses $G_i^{obs,ETO}$, $i \in \{1, \cdots, 4\}$ by adding some noise i.e.

$$G_{noisy,i}^{obs,ETO} = G_i^{obs,ETO} + \eta_i, \quad \eta_i \sim \mathcal{N}(0, \sigma_i), \quad \sigma_i = \left| G_i^{obs,ETO} \right|_{L^2(P)} \times 0.1\%, \quad (4.33)$$

where $\mathcal{N}(0, \sigma_i)$ is the Gaussian distribution with mean equal zero and standard deviation σ_i. The results of the inversion is illustrated in Tab 4.2 where p_i, p_i^{guess} and p_i^r are respectively the exact, guess and predicted model parameters associated to the synthetic response $G_i^{obs,ETO}$ for $r \in \{DG+AD, FE+FD\}$ and $i \in \{1, 2, 3, 4\}$. It shows that **DG + AD is more accurate compared to FE + FD to invert the noisy synthetic response of the ETO model**.

The results are also schematically presented in Fig 4.5. We plot in Fig 4.5(a), (d), (g) and (j) the observed voltammogram and the predicted voltammograms obtained using the approaches DG + AD and FE + FD respectively for the inversion experiment $i = 1, 2, 3, 4$. We respectively plot in Fig 4.5(b), (e), (h) and (k) the CPU time of the approaches DG + AD and FE + FD during the inversion of the observed data $G_i^{obs,ETO}$, $i = 1, 2, 3, 4$. Fig 4.5(b), (e), (h) and (k) show that **DG + AD is faster compared to FE + FD to invert the synthetic response of the ETO model**. We compute the absolute error on the dimensionless model parameters

using (4.31) and the absolute error on the dimensionless current as follows

$$E^r_{G,i} = 100\times\left|G^{obs,ETO}_{noisy,i} - G^r_i\right|_{L^2(P)}\times\left|G^{obs,ETO}_{noisy,i}\right|^{-1}_{L^2(P)}, \tag{4.34}$$

where G^r_i is the dimensionless current obtained from the predicted model parameter p^r_i for all $r \in \{DG+AD, FE+FD\}$ and $i \in \{1,2,3,4\}$. We then plot these absolute errors for each inversion method in Fig 4.5(c), (f), (i) and (l) respectively for the inversion experiment $i \in \{1,2,3,4\}$, which show that **DG + AD, compared to FE + FD, is more accurate to invert the noisy synthetic response of the ETO model**.

		K_0	D^+
$G^{obs,ETO}_{noisy,1}$	p_1^{guess}	17.316915142871846	8.074766022789072
	p_1^{DG+AD}	41.800960005493636	15.683814726637404
	p_1^{FE+FD}	16.832390259173525	16.138117391144760
	p_1	39.545563811966765	15.673817496388676
$G^{obs,ETO}_{noisy,2}$	p_2^{guess}	15.003689359617777	10.404183098085726
	p_2^{DG+AD}	9.963662931947123	7.733718482337124
	p_2^{FE+FD}	17.057427322383420	7.770485091972894
	p_2	10.114520194105989	7.791766414718039
$G^{obs,ETO}_{noisy,3}$	p_3^{guess}	49.943052408344080	2.986529885053503
	p_3^{DG+AD}	49.999998732770120	12.053813908235924
	p_3^{FE+FD}	49.628717764287515	3.588665800239379
	p_3	48.988630356887040	11.877524221053571
$G^{obs,ETO}_{noisy,4}$	p_4^{guess}	10.681195759200980	26.997401304716050
	p_4^{DG+AD}	4.547182492783893	4.083156095533337
	p_4^{FE+FD}	3.807230324644848	4.230128617271018
	p_4	4.517328151870212	4.047665369840588

Table 4.2: **Inversion of the ETO synthetic response with noise.** This table presents the model parameters, p_i, the initial guess, p_i^{guess}, the estimated parameters p_i^{DG+AD} and p_i^{FE+FD} respectively obtained with the approaches DG+AD and FE+FD, for the inversion of $G^{obs,ETO}_{noisy,i}$, $i = 1,\cdots,4$. It shows that DG + AD is more accurate compared to FE + FD.

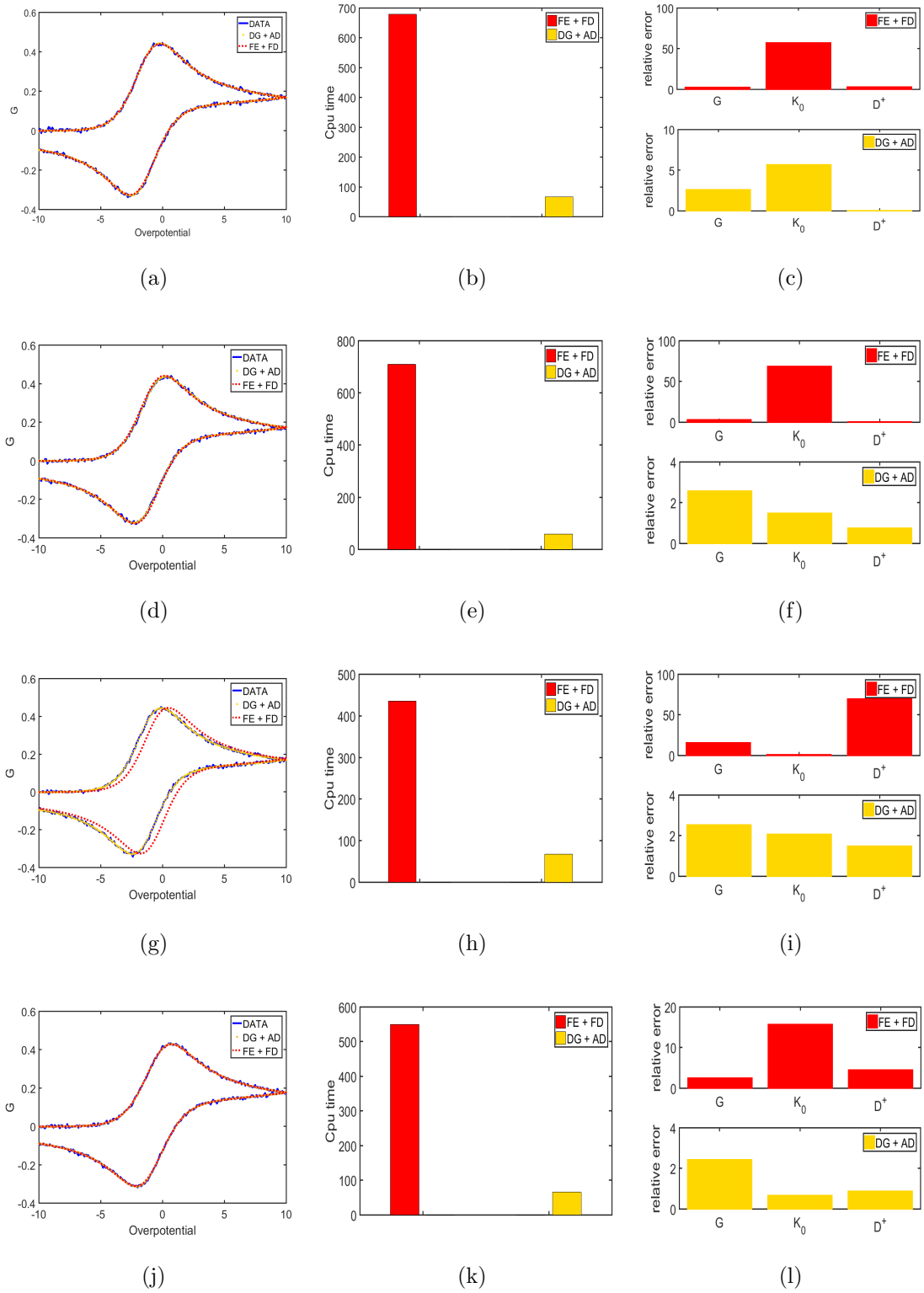


Figure 4.5: **Inversion of the ETO synthetic response with noise.** We plot in (a), (d), (g), (j) the observed voltammogram and the predicted voltammograms obtained using DG + AD and FE + FD respectively for the inversion experiment $i = 1, 2, 3, 4$. We respectively plot in (b), (e), (h), (k) the CPU time of DG + AD and FE + FD during the inversion of the observed data $G^{obs,ETO}_{noisy,i}$, $i = 1, 2, 3, 4$. As expected, note from (b), (e), (h), (k) that DG + AD is faster than FE + FD. We plot in (c), (f), (i), (l) the absolute error on the current and the components of the model parameter in each experiment of inversion $i = 1, 2, 3, 4$. Note from the different scales in (c), (f), (i), (l) that DG + AD is more accurate than FE + FD.

4.4 Numerical inversion of the EC' model

We have shown in Section 3.3 that the synthetic response of the EC' model can be computed in real time for a given value of the parameter

$$p^{EC'} = (p^{ETO}, K_r, D^-, K_A, D, \tilde{C}^0_{A^{total}}), \tag{4.35}$$

where p^{ETO} is defined by (4.22), K_r is the dimensionless catalytic rate constant, K_A is the dimensionless equilibrium rate constant, $\tilde{C}^0_{A^{total}}$ is the the dimensionless total concentration of the specie A at the initial time, D^- and D are respectively the dimensionless diffusion coefficient of the species B and A. We assume, as before for ETO model, that for the experimental data collected for EC model, the parameters $p^{known} = (\nu, C_m, T, P_1, P_2)$ are known. Therefore the inverse problem of EC' model is given by (4.2), with the model parameter $p = (K_0, D^+, K_r, D^-, K_A, D, \tilde{C}^0_{A^{total}})^T$, the final time $b = 2(P_2 - P_1)$, the function h and g, respectively derived from (3.85) and (3.88), take the following form

$$h(x, \dot{x}, p, t) = \dot{x} + \mathfrak{S}(t)x - N(X, K_r), \quad g(x\Big|_{t=0}, p) = x\Big|_{t=0} - x^0, \tag{4.36}$$

where x^0 is a vector depending only on $\tilde{C}^0_{A^{total}}$, and $x = (\alpha, \alpha^+, \beta^-, \beta)^T$ is a coupled component α, α^+, β^- and β respectively of the species $\mathbf{Q}$, $\mathbf{Q}^+, B$ and A in the DG finite space. Since the function h is non linear, then the gradient d_pF of the objective function F is computed by Algorithm 2 with the matrices $\mathcal{A}$, $\mathcal{B}$ and the non linear function $\mathfrak{N}(x, p)$ defined as follows

$$\mathcal{A} = -\mathfrak{S}(t), \quad \mathfrak{N}(x, p) = N(X, K_r), \quad \mathcal{B} = 0. \tag{4.37}$$

Algorithm 2 also requires the computation of the Jacobian $\partial_{x(0)}g$, the gradient $\partial_p f, \partial_p h$ and $\partial_p g$ respectively of the functions f, h and g.

4.4.1 Computation needed to invert EC' model

To simulate the inversion of the EC' model, we compute here the explicit form of the additional entities required. The Jacobian $\partial_x f(x,p,t)$ is given by

$$\partial_x f(x,p,t) = (G(x,p,t) - G^{obs})(K_f(t)Y_1, -K_b(t)Y_1, 0, 0), \tag{4.38}$$

where Y_1 is computed with (4.24). Since the function f depends exclusively on the parameter K_0, then the components of the gradient $\partial_p f(x,p,t)$ are given by

$$\partial_{K_0} f(x,p,t) = \frac{G(x,p,t)}{K_0}(G(x,p,t) - G^{obs})$$
$$\partial_{D^+} f = \partial_{K_r} f = \partial_{D^-} f = \partial_{K_A} f = \partial_D f = \partial_{\tilde{C}^0_{A^{total}}} f = 0.$$

Combining (3.23) and (4.36), we compute the entries of the gradient $\partial_p h$ as follow

$$\partial_{K_0} h(x,\dot{x},p,t) = \frac{G(x,p,t)}{K_0}(Y_2, -Y_2, 0, 0)^T, \quad \partial_{D^+} h(x,\dot{x},p,t) = (0, L_0^{s,\sigma_0}\alpha^+, 0, 0)^T$$
$$\partial_{K_A} h(x,\dot{x},p,t) = (0,0,Y_3,-Y_3)^T,\ \partial_D h(x,\dot{x},p,t) = (0,0,0,L_0^{s,\sigma_0}\beta)^T,\ \partial_{\tilde{C}^0_{A^{total}}} h = 0$$
$$\partial_{K_r} h(x,\dot{x},p,t) = -\frac{1}{K_r}N(x), \quad \partial_{D^-} h(x,\dot{x},p,t) = (0,0,L_0^{s,\sigma_0}\beta^-, 0)^T$$

where Y_3 is a n_T-vector given by $Y_3 = \beta^- - \beta$ and Y_2 given by (4.25).

Since the function g depends only on the the initial solution $x(0)$ and the dimensionless total concentration $\tilde{C}^0_{A^{total}}$ of the specie A for the EC' model, then the Jacobian $\partial_{x(0)} g$ and the gradient $\partial_p g$ are given by

$$\partial_{D^+} g = \partial_{K_r} g = \partial_{D^-} g = \partial_{K_A} g = \partial_D g = \partial_{K_0} g = 0$$
$$\partial_{\tilde{C}^0_{A^{total}}} g = -\frac{1}{\tilde{C}^0_{A^{total}}}(0,0,\beta^-(t=0),\beta(t=0))^T, \qquad \partial_{x(0)} g = \mathcal{I}.$$

Once the adjoint equation is solved for the Lagrangian coefficient λ, then the

entries of the gradient d_pF of the objective function are given by

$$d_{K_0}F(x,p) = \int_0^{2t_\lambda} \underbrace{\frac{G(x,p,t)}{K_0}\left[(G(x,p,t)-G^{obs})+(\lambda_1-\lambda_2)Y_2\right]}_{f_1(x,p,t)}\,dt$$

$$d_{D^+}F(x,p) = \int_0^{2t_\lambda} \underbrace{\lambda_2 L_0^{s,\sigma_0}\alpha^+}_{f_2(x,p,t)}\,dt, \quad d_{D^-}F(x,p) = \int_0^{2t_\lambda} \underbrace{\lambda_3 L_0^{s,\sigma_0}\beta^-}_{f_4(x,p,t)}\,dt,$$

$$d_{K_r}F(x,p) = \int_0^{2t_\lambda} \underbrace{-(2\lambda_1-2\lambda_2-\lambda_3)N_1(x)}_{f_3(x,p,t)}\,dt, \quad d_{K_A}F(x,p) = \int_0^{2t_\lambda} \underbrace{(\lambda_3-\lambda_4)Y_3}_{f_5(x,p,t)}\,dt$$

$$d_DF(x,p) = \int_0^{2t_\lambda} \underbrace{\lambda_4 L_0^{s,\sigma_0}\beta}_{f_6(x,p,t)}\,dt, \quad d_{\tilde{C}^0_{A^{total}}}F(x,p) = -\frac{1}{\tilde{C}^0_{A^{total}}}(\lambda_3\beta^- + \lambda_4\beta)\Big|_{t=0},$$

where $\lambda^T = (\lambda_1, \lambda_2, \lambda_3, \lambda_4)$ and λ_i is $1 \times n_T$ vector for all $i = 1, 2, 3, 4$.

4.4.2 Numerical experiments on the inversion of the synthetic response of the EC' model

The purpose in this section is to first validate the adjoint method for the computation of the gradient of the objective function in the case of the EC' model. Finally, we investigate the performance of the DG , Impl and the adjoint method to invert the synthetic data from EC' model, considered as observed data. Unless stated, we consider the synthetic dimensionless current associated to the parameter $p^{obs} = (8, 1, 10, 5, 10, 5, 1)^T$, as our observed data G^{obs} and the model parameters are bounded as follows

$$K_0, D^+, D^-, K_A, D \in [0.1, 20], \quad K_r \in [0.1, 10^4] \text{and } \tilde{C}^0_{A^{total}} \in [0.1, 2]. \tag{4.39}$$

In order to make the optimization method more efficient, we introduce the normalized parameters $\hat{p}_i = (p_i - a_i)(b_i - a_i)^{-1}, i \in \{1, \cdots, 7\}$, for all parameter $p_i \in [a_i, b_i], i \in \{1, \cdots, 7\}$. Thus, we have $\hat{p}_i \in [0, 1]$ for all $i = 1, \cdots, 7$ and the entries of the gradient $d_{\hat{p}}F$ of the objective function F are $d_{\hat{p}_i}F = (b_i - a_i)d_{p_i}F, i \in \{1, \cdots, 7\}$. Let us respectively denote $\hat{K}_0^{obs}, \hat{D}_{obs}^+, \hat{K}_r^{obs}, \hat{D}_{obs}^-, \hat{K}_A^{obs}, \hat{D}^{obs}$ and $\hat{C}^{0,obs}_{A^{total}}$ the normalized values of the default parameters $K_0^{obs}, D_{obs}^+, K_r^{obs}, D_{obs}^-, K_A^{obs}, D^{obs}$ and $\tilde{C}^{0,obs}_{A^{total}}$.

Numerical experiment 1: test adjoint method for EC model

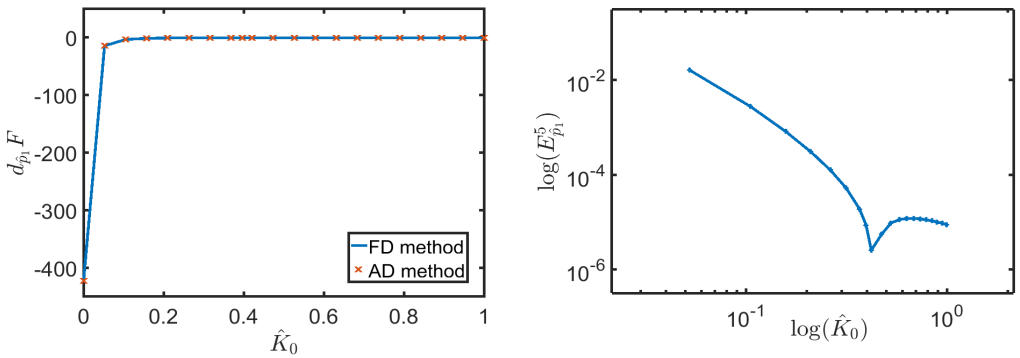
To validate the adjoint method for the EC' model, we compare the entries of gradient $d_{\hat{p}}F$, of the objective function F, obtained by using Algorithm 2 and (4.7) i.e. the centred the finite difference formula with $\epsilon = 10^{-4}$. We focus here on the first entry, denoted $d^n_{\hat{p}_1}F$, of the gradient $d_{\hat{p}}F$ at all the point

$$\hat{p}^n = (\hat{K}_0^n = n \times (b_1 - a_1)^{-1}, \hat{D}^+_{obs}, \hat{K}^{obs}_r, \hat{D}^-_{obs}, \hat{K}^{obs}_A, \hat{D}^{obs}, \hat{C}^{0,obs}_{A^{total}}),$$

for all $n \in \{0, \cdots, 20\}$. Then, we compute the difference

$$E^5_{\hat{p}^n_1} = \left| d^{n,1}_{\hat{p}_1}F - d^{n,2}_{\hat{p}_1}F \right|, \ \forall n \in \{0, \cdots, 20\}, \tag{4.40}$$

where $d^{n,1}_{\hat{p}_1}F$ and $d^{n,2}_{\hat{p}_1}F$ are respectively the first entry of the gradient $d_{\hat{p}}F$ computed with Algorithm 2 and (4.7). We plot in Fig 4.6(a) $d^{n,r}_{\hat{p}_1}F$ against the normalized parameter $\hat{K}^n_0$ for all $r = 1, 2$. We plot in Fig 4.6(b) the difference $\log(E^5_{\hat{p}^n_i})$ as a function of $\log(\hat{K}^n_0)$. We can conclude from Fig 4.6(b) that $d^{n,1}_{\hat{p}_1} \approx d^{n,2}_{\hat{p}_1}$. The same procedure is followed in Section A.5.2 to compare the remaining entries of the gradient $d_{\hat{p}}F$ using the adjoint method or the centred finite element method (the result of this comparison is shown in Fig A.2 - Fig A.7).

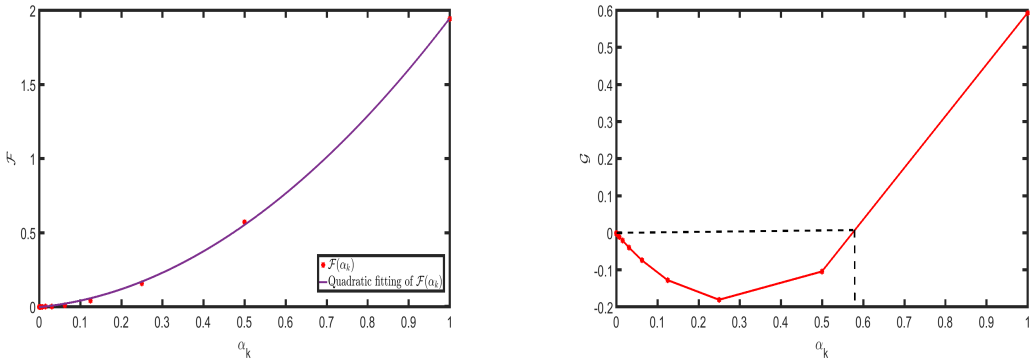


(a) (b)

Figure 4.6: **Comparison of the gradient using adjoint and finite difference method for the EC' model.** In (a) we plot the total derivative $d_{\hat{p}_1}F$, compute with the adjoint and finite difference method, as a function of the parameter $\hat{K}_0$. In (b) we plot the logarithm of the difference $E^5_{\hat{p}^n_1}$ as a function of the logarithm of $\hat{K}^n_0$.

As in the case of ETO model in Section 4.3.2, to reinforce the validation of the ad-

joint method and its implementation for the EC' model, we check if (4.29) and (4.30) hold for a given model parameter p. To this end, we set $p = (9.95, 9.95, 9.95, 9.95, 9.95, 9.95, 1)$ and compute $d_pF(p)$ with Algorithm 2. By considering $\mathcal{D} = -d_pF(p) \parallel d_pF(p) \parallel^{-1}$, we compute $\mathcal{F}(\alpha^n)$ and $\mathcal{G}(\alpha^n)$ for all $\alpha^n = 2^{-n},\ n \in \{0, \cdots, 20\}$. In Fig 4.7(a), we plot $\mathcal{F}(\alpha^n)$ against α^n and fit the result with MATLAB quadratic function. Note from Fig 4.7(a) shows that (4.29) holds for $b_0 = 1$. We also plot in Fig 4.7(b) $\mathcal{G}(\alpha^n)$ against α^n, which shows that (4.30) holds for $b_1 = 0.5$.



(a) Function $\mathcal{F}$ (b) Function $\mathcal{G}$

Figure 4.7: **Test of Taylor expansion and gradient descent property using the adjoint method for EC' model.** In (a), we plot $\mathcal{F}(\alpha^n)$ against α^n and fit the result with MATLAB's quadratic function. It shows that Taylor expansion holds for $b_0 = 1$. In (b), we plot $\mathcal{G}(\alpha^n)$ against α^n, which shows that the gradient descent property holds for $b_1 = 0.5$.

Now that the adjoint method for the EC' model has been validated, we focus on the inversion of the noiseless and noisy synthetic responses of EC' model, in the following numerical experiments.

Numerical experiment 2: Efficiency of the combination of DG, Impl and adjoint method for the inversion of the synthetic response of EC'.

To simulate the performance of DG method against `pdepe`, for the inversion of EC' synthetic response, we invert a sample of four synthetic responses of the EC' model. These synthetic responses $G_i^{obs,EC'}$ are associated to the model parameters p^i generated randomly such that (4.39) holds for all $i \in \{1, 2, 3, 4\}$. First the approaches FE+FD and DG+AD are used to supply the objective function F and its gradient d_pF to the optimisation code `fmincon` of MATLAB under the default options. For

an extensive description on the `fmincon` Interior Point Algorithm, see [45, 46, 218]. The starter point of `fmincon` is also generated randomly such that (4.39) holds.

The optimization results are summarized in Tab 4.3 and show that DG+AD is more accurate in terms of the estimated model parameters. The same structure, as for ETO model (see Fig 4.4), is used to schematically display the results in Fig 4.8. Fig 4.8 (a), (d), (g) and (j) show that the observed voltammogram is well fitted with DG+AD, but not with FE+FD. The different scales in Fig 4.8 (c), (f), (i) and (l) show that the model parameters are more accurately predicted with DG+AD.

		K_0	D^+	K_r	D^-	K_A	D	$\tilde{C}^0_{A^total}$
$G_1^{obs,EC'}$	p_1^{guess}	2.1024	15.3597	941.6366	6.7825	2.0334	4.4809	1.0746
	p_1^{DG+AD}	13.3208	6.6453	3850.75	1.8150	15.4974	6.1808	0.2763
	p_1^{FE+FD}	1.4231	15.2294	941.6364	4.9753	0.1096	3.9042	0.2147
	p_1	13.3339	6.6445	3847.92	1.8031	15.5743	6.1812	0.2764
$G_2^{obs,EC'}$	p_2^{guess}	19.6456	2.6058	8525.04	9.0522	19.9785	19.5280	1.4852
	p_2^{DG+AD}	19.6725	5.2212	8525.03	0.6973	3.2612	4.2792	0.6273
	p_2^{FE+FD}	14.7957	10.8584	8525.85	0.1206	18.9801	12.7435	1.0873
	p_2	18.4321	5.2203	9059.63	0.6969	3.3286	4.3159	0.6256
$G_3^{obs,EC'}$	p_3^{guess}	19.1343	7.7158	7804.92	2.9023	14.4371	12.7734	1.10922
	p_3^{DG+AD}	8.1605	4.4636	7804.77	17.9760	17.9260	5.6951	1.9997
	p_3^{FE+FD}	19.1343	7.7158	7804.92	2.9023	14.4371	12.7733	1.1092
	p_3	8.0274	4.7231	9736.28	19.6732	18.6818	7.4633	1.8723
$G_4^{obs,EC'}$	p_4^{guess}	13.7984	5.4695	1167.44	11.4793	1.1074	10.8697	1.6364
	p_4^{DG+AD}	3.8453	19.4187	7692.33	14.8909	7.6622	8.3843	1.7080
	p_4^{FE+FD}	13.7984	5.4695	1167.44	11.4793	1.1074	10.8697	1.6364
	p_4	3.8452	19.4348	7696.39	14.8813	7.6564	8.3756	1.7086

Table 4.3: **Results of inversion of the synthetic response of the EC' model with FE+FD and DG+AD combined with `fmincon` MATLAB code under the default settings.** This table presents the model parameters p, the initial guess p^{guess}, the estimated parameters p^{DG+AD} and p^{FE+FD} respectively obtained with the DG+AD and FE+FD approaches.

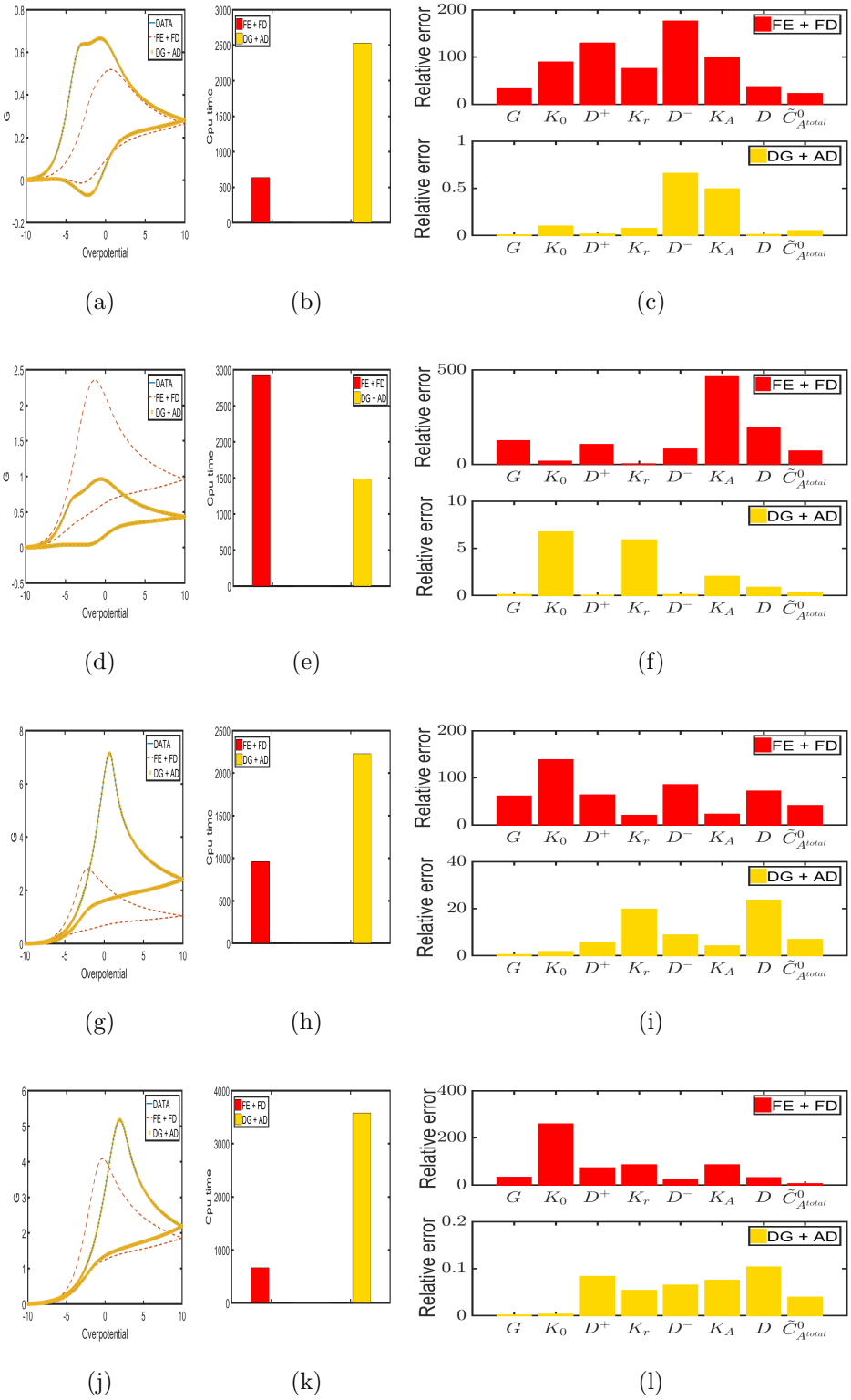


Figure 4.8: **Inversion of the synthetic response of the EC' model with FE+FD and DG+AD combined with `fmincon` MATLAB code under the default settings.** We plot in (a), (d), (g), (j) the observed voltammogram and the predicted voltammograms obtained using DG + AD and FE + FD respectively for the inversion experiment $i = 1, 2, 3, 4$. We respectively plot in (b), (e), (h), (k) the CPU time of DG + AD and FE + FD during the inversion of the observed data $G_i^{obs,EC'}$, $i = 1, 2, 3, 4$. We plot in (c), (f), (i), (l) the absolute error on the current and the components of the model parameter in each experiment of inversion $i = 1, 2, 3, 4$. Note from the different scales in (c), (f), (i), (l) that DG + AD is more accurate than FE + FD.

Secondly, we use DG+FD to supply the objective function F and its gradient d_pF to `fmincon` with default settings and compare the results to the one previously obtained with DG+AD. The results of DG+FD and DG+AD are summarized in Tab 4.4 and schematically displayed in Fig 4.9. In this case, Fig 4.9 (a), (d), (g) and (j) show that the observed voltammogram is well fitted with DG+AD or DG+FD; and as expected Fig 4.9 (b), (e), (h) and (k) show that DG+AD is less expensive in term of CPU time compared to DG+FD. Since FE+FD can't even fit the observed voltammogram, we can conclude that the better conservative property of the DG is very important for the inversion.

		K_0	D^+	K_r	D^-	K_A	D	$\tilde{C}^0_{A^t otal}$
$G_1^{obs,EC'}$	p_1^{guess}	2.1024	15.3597	941.6366	6.7825	2.0334	4.4809	1.0746
	p_1^{DG+AD}	13.3208	6.6453	3850.75	1.8150	15.4974	6.1808	0.2763
	p_1^{DG+FD}	13.3204	6.6453	3850.89	1.8149	15.4983	6.1810	0.2763
	p_1	13.3339	6.6445	3847.92	1.8031	15.5743	6.1812	0.2764
$G_2^{obs,EC'}$	p_2^{guess}	19.6456	2.6058	8525.04	9.0522	19.9785	19.5280	1.4852
	p_2^{DG+AD}	19.6725	5.2212	8525.03	0.6973	3.2612	4.2792	0.6273
	p_2^{DG+FD}	19.6724	5.2212	8525.03	0.6974	3.2612	4.2794	0.6273
	p_2	18.4321	5.2203	9059.63	0.6969	3.3286	4.3159	0.6256
$G_3^{obs,EC'}$	p_3^{guess}	19.1343	7.7158	7804.92	2.9023	14.4371	12.7734	1.10922
	p_3^{DG+AD}	8.1605	4.4636	7804.77	17.9760	17.9260	5.6951	1.9997
	p_3^{DG+FD}	8.0270	4.7231	9720.66	19.6189	18.4795	7.4008	1.8762
	p_3	8.0274	4.7231	9736.28	19.6732	18.6818	7.4633	1.8723
$G_4^{obs,EC'}$	p_4^{guess}	13.7984	5.4695	1167.44	11.4793	1.1074	10.8697	1.6364
	p_4^{DG+AD}	3.8453	19.4187	7692.33	14.8909	7.6622	8.3843	1.7080
	p_4^{DG+FD}	3.8453	19.4153	7693.29	14.8947	7.6842	8.3999	1.7073
	p_4	3.8452	19.4348	7696.39	14.8813	7.6564	8.3756	1.7086

Table 4.4: **Results of inversion of the synthetic response of the EC' model with DG+FD and DG+AD combined with fmincon MATLAB code under the default settings.** This table presents the model parameters p, the initial guess p^{guess}, the estimated parameters p^{DG+AD} and p^{DG+FD} respectively obtained with the DG+AD and DG+FD approaches.

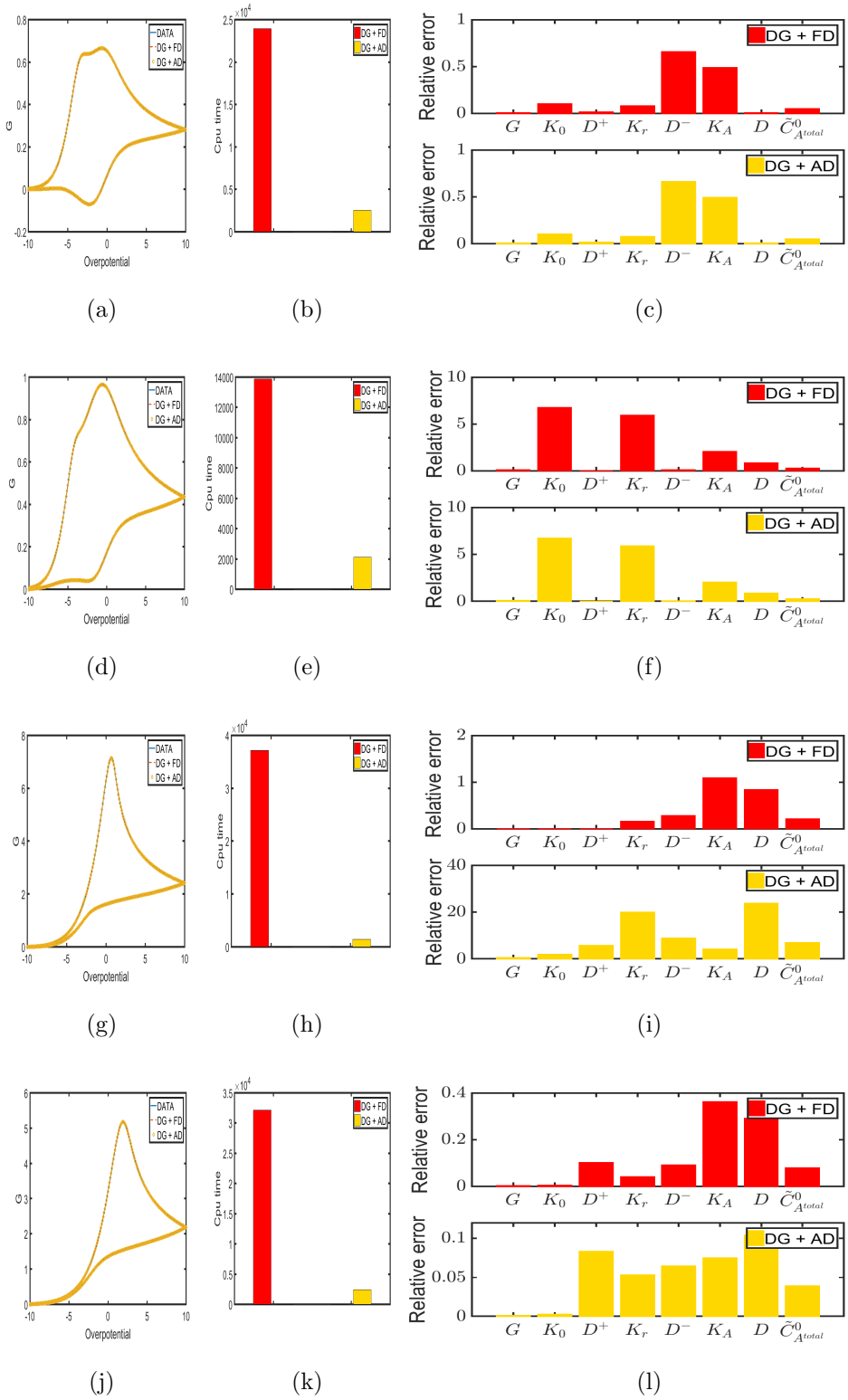


Figure 4.9: **Inversion of the synthetic response of the EC' model with DG+FD and DG+AD combined with fmincon MATLAB code under the default settings.** We plot in (a), (d), (g), (j) the observed voltammogram and the predicted voltammograms obtained using DG + AD and DG + FD respectively for the inversion experiment $i = 1, 2, 3, 4$. We respectively plot in (b), (e), (h), (k) the CPU time of DG + AD and DG + FD during the inversion of the observed data $G_i^{obs,EC'}$, $i = 1, 2, 3, 4$. We plot in (c), (f), (i), (l) the absolute error on the current and the components of the model parameter in each experiment of inversion $i = 1, 2, 3, 4$. Note from the different scales in (b), (e), (h), (k) that DG + AD is less expensive than DG + FD.

We finally invert the same synthetic responses using the approaches FE+FD and DG+AD to supply the objective function F and its gradient d_pF to fmincon subject to the termination tolerance $TolFun = 10^{-4}$ on the function F, Termination tolerance $TolX = 10^{-4}$ on the parameter p step size factor for finite differences $H_{FD} = 10^{-2}$, which by default are $TolX = TolFun = 10^{-6}$ and $H_{FD} = \sqrt{eps}$. The results are presented in Fig 4.10 and Tab 4.5. It shows that by considering these additional constraints for fmincon, we can improve the outcomes of FE+FD but it still not good as the the outcomes of DG+AD.

		K_0	D^+	K_r	D^-	K_A	D	$\tilde{C}^0_{A^total}$
$G_1^{obs,EC'}$	p_1^{guess}	2.1024	15.3597	941.6366	6.7825	2.0334	4.4809	1.0746
	p_1^{DG+AD}	12.4445	6.6873	4092.37	2.5124	11.9328	6.2738	0.2675
	p_1^{FE+FD}	9.5971	9.0782	5124.49	7.8814	9.8545	8.2282	0.2202
	p_1	13.3339	6.6445	3847.92	1.8031	15.5743	6.1812	0.2764
$G_2^{obs,EC'}$	p_2^{guess}	19.6456	2.6058	8525.04	9.0522	19.9785	19.5280	1.4852
	p_2^{DG+AD}	17.81174	5.2677	8993.81	0.6925	3.2663	4.1405	0.6341
	p_2^{FE+FD}	15.3093	5.8324	7206.75	2.8389	4.0656	17.8076	0.3434
	p_2	18.4321	5.2203	9059.63	0.6969	3.3286	4.3159	0.6256
$G_3^{obs,EC'}$	p_3^{guess}	19.1343	7.7158	7804.92	2.9023	14.4371	12.7734	1.10922
	p_3^{DG+AD}	8.0240	4.7114	9461.51	18.8606	15.3099	6.4511	1.9373
	p_3^{FE+FD}	8.0744	3.2034	7965.15	18.4274	14.6555	15.6517	1.6918
	p_3	8.0274	4.7231	9736.28	19.6732	18.6818	7.4633	1.8723
$G_4^{obs,EC'}$	p_4^{guess}	13.7984	5.4695	1167.44	11.4793	1.1074	10.8697	1.6364
	p_4^{DG+AD}	3.8499	18.6840	7544.45	15.0992	9.17862	9.4172	1.6653
	p_4^{FE+FD}	3.7883	8.6808	4044.60	15.4952	12.6857	13.9111	1.5168
	p_4	3.8452	19.4348	7696.39	14.8813	7.6564	8.3756	1.7086

Table 4.5: **Results of inversion of the synthetic response of the EC' model with FE+FD and DG+AD combined with fmincon MATLAB code under the non default settings.** This table presents the model parameters p, the initial guess p^{guess}, the estimated parameters p^{DG+AD} and p^{FE+FD} respectively obtained with the DG+AD and FE+FD approaches.

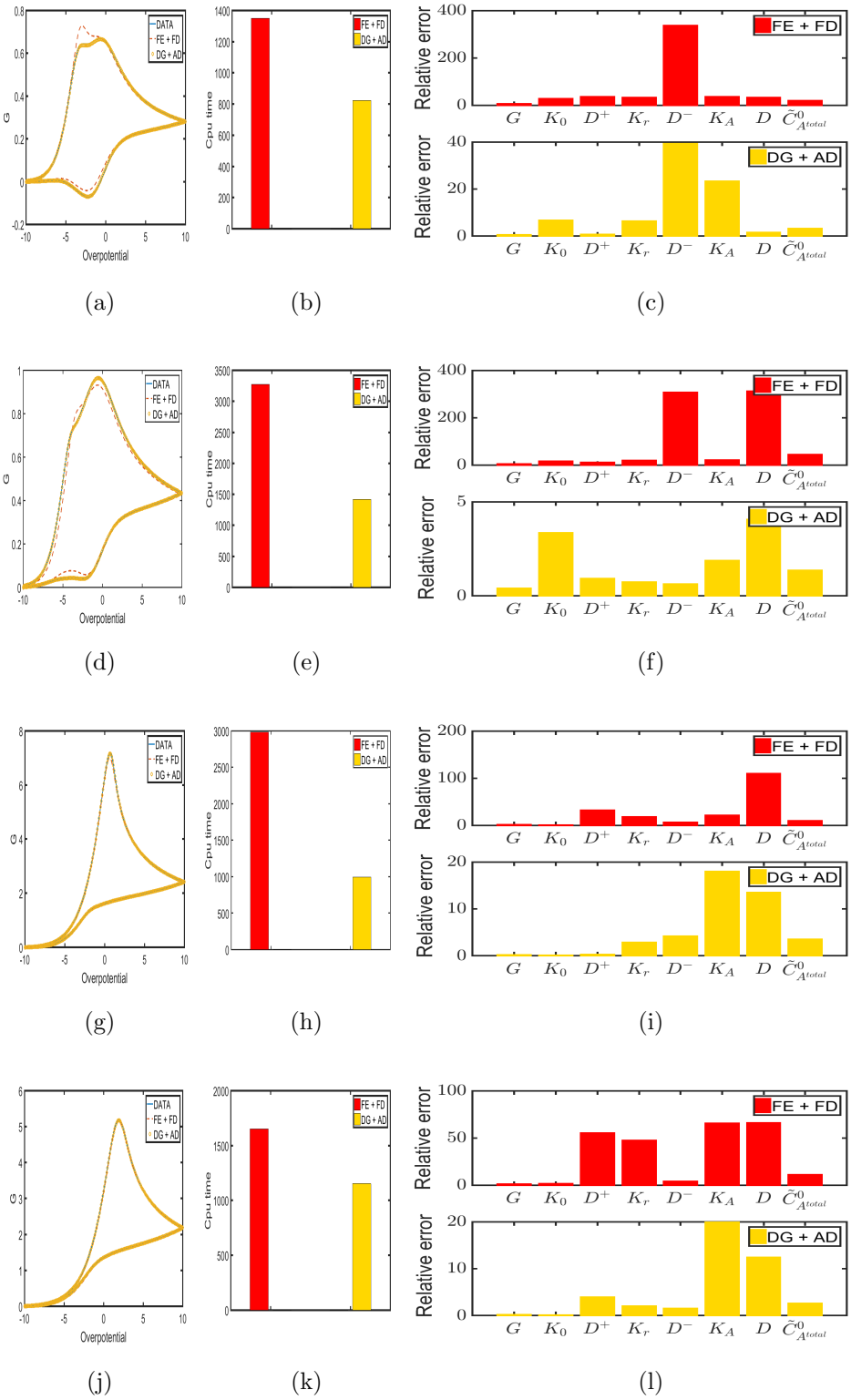


Figure 4.10: **Inversion of the synthetic response of EC' model with FE+FD and DG+AD combined with fmincon subject to non default options.** We plot in (a), (d), (g), (j) the observed voltammogram and the predicted voltammograms obtained using DG + AD and FE + FD respectively for the inversion experiment $i = 1, 2, 3, 4$. We respectively plot in (b), (e), (h), (k) the CPU time of DG + AD and FE + FD during the inversion of the observed data $G_i^{obs,EC'}$, $i = 1, 2, 3, 4$. We plot in (c), (f), (i), (l) the absolute error on the current and the components of the model parameter in each experiment of inversion $i = 1, 2, 3, 4$. these figures show that DG + AD is more efficient than FE + FD.

Numerical experiment 3: Efficiency of the combination of DG, Impl and adjoint method for the inversion of the synthetic response of ETO with some additional random noise.

As for the ETO model case, we defined the noisy synthetic responses $G^{obs,EC'}_{noisy,i}$ by adding some Gaussian random noise to the synthetic response $G^{obs,EC'}_{i}$ for all $i \in \{1,2,3,4\}$. The noisy synthetic response is then invert with FE+FD and DG+AD approaches to supply the objective function F and its gradient d_pF to fmincon subject to the default option.

		K_0	D^+	K_r	D^-	K_A	D	$\tilde{C}^0_{A^total}$
$G^{obs,EC'}_{noisy,1}$	p^{guess}	2.1024	15.3597	941.6366	6.7825	2.0334	4.4809	1.0746
	p^{DG+AD}	14.1194	6.5525	3700.24	1.4416	16.9973	6.5054	0.2772
	p^{FE+FD}	1.5691	15.2600	941.636	5.0127	0.1096	4.0031	0.2082
	p	13.3339	6.6445	3847.92	1.8031	15.5743	6.1812	0.2764
$G^{obs,EC'}_{noisy,2}$	p^{guess}	19.6456	2.6058	8525.04	9.0522	19.9785	19.5280	1.4852
	p^{DG+AD}	19.2305	5.2876	8525.00	0.7005	3.1934	4.2764	0.6271
	p^{FE+FD}	6.0807	1.9939	8524.96	0.1447	19.8583	19.4107	1.0530
	p	18.4321	5.2203	9059.63	0.6969	3.3286	4.3159	0.6256
$G^{obs,EC'}_{noisy,3}$	p^{guess}	19.1343	7.7158	7804.92	2.9023	14.4371	12.7734	1.1092
	p^{DG+AD}	8.2238	4.6764	7805.51	18.4184	18.6152	5.3227	1.9985
	p^{FE+FD}	19.1343	7.7158	7804.92	2.9023	14.4371	12.7734	1.1092
	p	8.0274	4.7231	9736.28	19.6732	18.6818	7.4633	1.8723
$G^{obs,EC'}_{noisy,4}$	p^{guess}	13.7984	5.4695	1167.44	11.4793	1.1074	10.8697	1.6364
	p^{DG+AD}	3.8130	15.5459	7732.76	19.0324	18.0103	17.2865	1.3716
	p^{FE+FD}	14.0963	5.8524	1167.45	12.6033	11.6313	12.1876	1.6920
	p	3.8452	19.4348	7696.39	14.8813	7.6564	8.3756	1.7086

Table 4.6: **Inversion of the synthetic response of EC' model plus additional random noise with the FE+FD and DG+AD method combined with the fmincon MATLAB code under the default settings.** This table presents the model parameters p, the initial guess p^{guess}, the estimated parameters p^{DG+AD} and p^{FE+FD} respectively obtained with the DG+AD and FE+FD approaches.

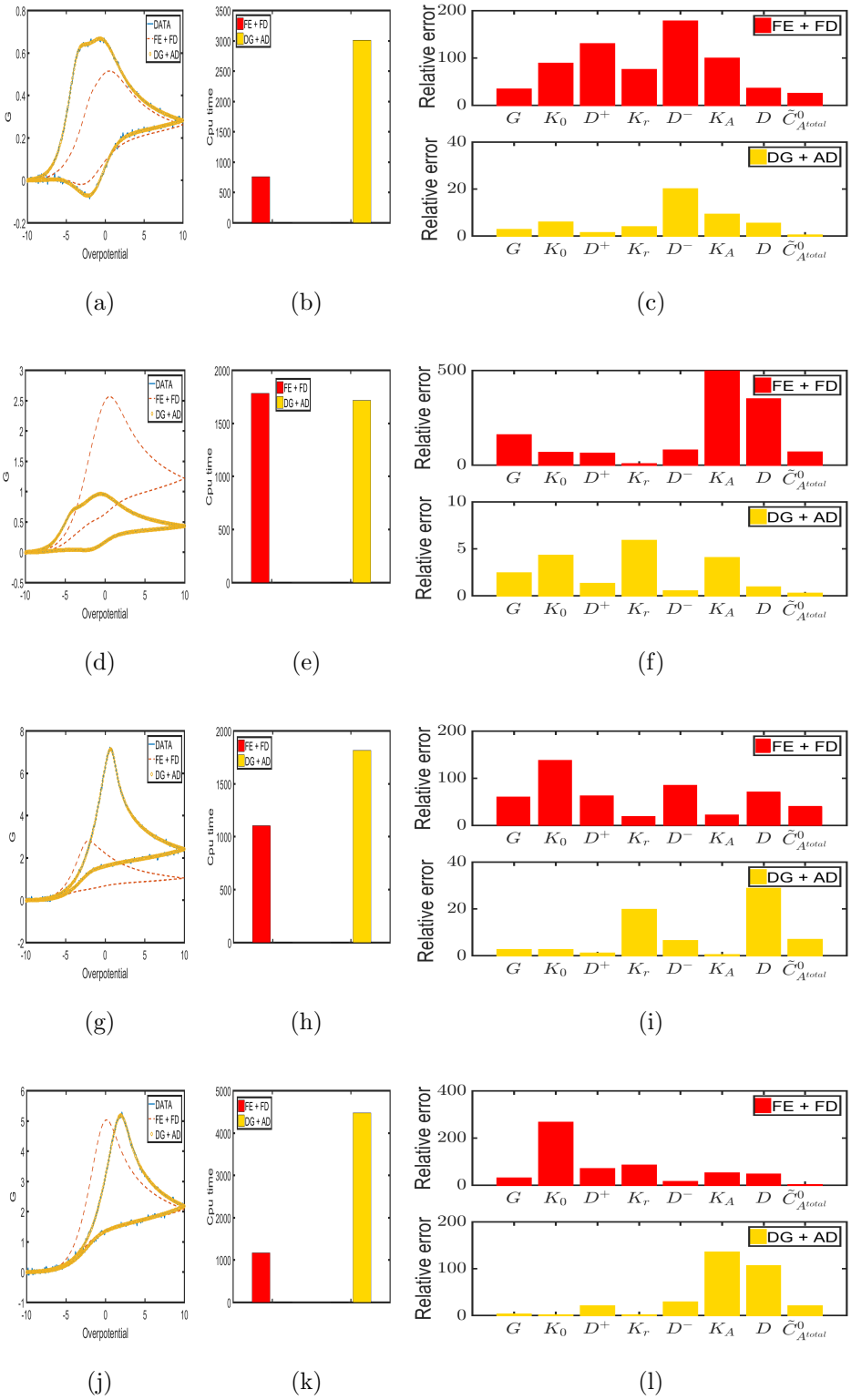


Figure 4.11: **Inversion of the synthetic response of EC' model plus additional random noise with FE+FD and DG+AD combined with fmincon MATLAB code under the default settings.** We plot in (a), (d), (g), (j) the observed voltammogram and the predicted voltammograms obtained using DG + AD and FE + FD respectively for the inversion experiment $i = 1, 2, 3, 4$. We respectively plot in (b), (e), (h), (k) the CPU time of DG + AD and FE + FD during the inversion of the observed data $G_i^{obs,EC'}$, $i = 1, 2, 3, 4$. We plot in (c), (f), (i), (l) the absolute error on the current and the components of the model parameter in each experiment of inversion $i = 1, 2, 3, 4$. Note from (a), (d), (g), (j) that DG + AD gives a good fitting of the noisy observed voltammogram while FE + FD does not.

The results Fig 4.9 and Tab 4.4 show that DG+AD gives a good fit of the dimensionless noisy current, while FE+FD fitting is very bad. The relative error of the estimated model parameters obtained with DG+AD, shows that some improvement needs to be done.

4.5 Summary

We develop point by point in this chapter an inversion solver for the measurements of the dimensionless current associated to ETO and EC' model. This inversion solver combined

- DG method for the space discretization,
- implicit Euler method for the time discretization,
- the adjoint method for the computation of the gradient,
- and gradient descent or fmincon Interior Point Algorithm for the minimization.

Schematically this can be represented as follows

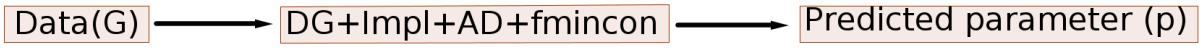


Figure 4.12: Inversion model 1.

In order to validate the good performance of our fit-for-purpose inversion solver, we investigate another inversion model, based only on the MATLAB PDEs solver *pdepe* and the gradient implicitly compute with fmincon MATLAB code. This MATLAB inversion model can be schematically represent as follow

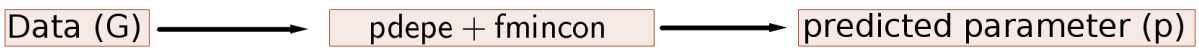


Figure 4.13: Inversion model 2

The comparison of the results of several numerical experiments shows that the inversion model Fig 4.12 is more fast, is more accurate and has better stability

property for the inversion of the noisy or noiseless synthetic response of ETO and EC' model. But it's still need some improvements in the case of EC' model, since the relative error on the estimated model parameters can be very high ($> 25\%$) for a good fit of the noiseless or noisy synthetic response of the EC' model.

To improve the inversion model Fig 4.12 for the EC' model, one can use for example the Tikhonov regularisation method [197, 171, 93, 122] or another optimisation algorithm (stochastic or deterministic).

Chapter 5

Two or three dimensional discontinuous Galerkin method for flow and transport in porous media

Contents

This chapter is devoted to the approximation of the model of flow and transport in porous media. We focus on the so-called method of lines in which the evolution problem is first semidiscretized in space yielding a system of coupled ordinary differential equations (ODEs) which is then discretized in time. Specifically, we consider DG semidiscretization together with standard time discretization method (such as Implicit Euler, ETD, EXPR and ROCK2 methods, see Chapter 2) and uniform time step on the solution domain. To that end, we first define in Section 5.1 the concept of flow and transport in porous media. Then we review, in Section 5.2, the two or three dimensions DG semidiscretization applied to DAREs. Finally, we use

the proposed numerical method to solve different models of flow and transport in porous media. Most materials for the DG semidretization schemes, presented in this chapter, are drawn from Alexandre Ern and Daniele Antonio Di Pietro [80]. Note that, there is a large literature showing the successful application of the DG method to the DAREs, see for example [98, 208, 133, 221, 18, 13, 57, 195, 116, 43].

The novel contributions in this chapter are

- the combination of ideas in [80, 125], to efficiently assemble the matrices that arise from the DG semidiscretization (i.e. mass and stiffness matrices),
- the comparison of the performance of the DG method combined with the time solvers (such as ETD1, ETD2 and EXPR) described in Section 2.3 to solve the DAREs.

5.1 Introduction to flow and transport in porous media

A porous medium is a material which contains void space called pores allowing a fluid (liquid or gas) to flow through. Many natural substances such as rocks, soils, biological tissue (e.g. bones), and man-made materials such as cements, foams and ceramics can be considered as porous media. A porous medium is characterised by its porosity, permeability as well as the properties of its constituents (solid matrix and fluid). We focus here on a porous medium which contains different types of rocks, called a geological reservoir. If the reservoir contains oil or gas, it is called an oil reservoir or hydrocarbon reservoir, if it contains water it is called a groundwater reservoir and if it contains heat it is called a geothermal reservoir.

Porosity, ϕ, is a measure of the empty spaces in a material, and is given by fraction of the volume of voids over the total volume i.e.

$$\phi = \frac{\textbf{Volume of pores}}{\textbf{Total Volume of porous medium}}.$$

Therefore the porosity is between 0 and 1, or as a percentage between 0 and 100%.

Several methods can be used to measure porosity : tomography method using industrial CT scanning, water saturation method, water evaporation method, gas expansion method [89], thermoporosimetry [40], cryoporometry [167], etc. Tab 5.1 shows the average of the porosity of some important rock types.

Material	Porosity (%)	Material	Porosity (%)
Sand, coarse	39	Siltstone	35
Sand, medium	39	Claystone	43
Sand, fine	43	Shale	6
Gravel, Coarse	28	Loess	49
Gravel, medium	32	Peat	92
Gravel, fine	34	Schist	38

Table 5.1: Average of the porosity of some sand and gravel [182].

The permeability, $\mathbf{k}$, in the standard international unit $[m^2]$ or in practical unit [Darcy], is the measure of the ability of a porous media to allow fluids to pass through. The unit [Darcy] is named after the French engineer Henry Darcy, who made several important contributions to hydraulics and we have 1Darcy $\approx 10^{-12}m^2$. In three dimensions, the permeability of anisotropic porous medium is a full tensor given by

$$\mathbf{k} = \begin{pmatrix} k_{xx} & k_{xy} & k_{xz} \\ k_{yx} & k_{yy} & k_{yz} \\ k_{zx} & k_{zy} & k_{zz} \end{pmatrix}. \tag{5.1}$$

If the porous medium is isotropic, then the permeability is reduced to $\mathbf{k} = k\mathbf{I}_3$, where $\mathbf{I}_3$ is the 3×3 identity tensor and $k \in \mathbb{R}$.

In hydro-geology, the permeability is combined with the fluid properties viscosity μ, the density ρ, and the constant of gravity g to form the hydraulic conductivity $\mathbf{K}$ in $[ms^{-1}]$ given by

$$\mathbf{K} = \frac{\rho g}{\mu}\mathbf{k}. \tag{5.2}$$

Tab 5.2 illustrates the range of the hydraulic conductivities (assuming pure water at room conditions) for some geological media.

	Rocks	Hydraulic conductivity(ms^{-1})
Unconsolidated sedimentary	Gravel	3×10^{-4} to 3×10^{-2}
	Medium sand	9×10^{-7} to 6×10^{-4}
	Silt, loess	1×10^{-9} to 2×10^{-5}
	Till	1×10^{-12} to 2×10^{-6}
	Clay	1×10^{-11} to 5×10^{-9}
	Unweathered marine clay	8×10^{-13} to 2×10^{-9}
Sedimentary rocks	Karst limestone	1×10^{-6} to 2×10^{-2}
	Limestone and dolomite	1×10^{-9} to 6×10^{-6}
	Sandstone	3×10^{-10} to 6×10^{-6}
	Shale	1×10^{-13} to 2×10^{-9}
Crystalline rocks	Permeable basalt	4×10^{-7} to 2×10^{-2}
	Fractured igneous and metamorphic	8×10^{-9} to 3×10^{-4}
	Basalt	2×10^{-11} to 4×10^{-7}
	Unfractured igneous and metamorphic	3×10^{-14} to 2×10^{-10}
	Weathered grantie	3×10^{-6} to 5×10^{-5}

Table 5.2: Hydraulic conductivity of the water in different porous media [182].

5.1.1 Darcy's law

In 1856, Henry Darcy investigated the flow of water in a vertical, saturated, homogeneous sand filter, see Fig 5.1, in connection with Dijon city's fountains. From his experiments, varying the length and diameter of the column, the porous material in it, and the water levels h_3 and h_4, respectively in the inflow and outflow reservoirs of the column, he concluded that the rate of flow Q (volume of water passing through the porous material per unit time) through a sand column of length L and constant cross-sectional area is:

- proportional to the cross-sectional area A of the column,
- proportional to the difference $h_3 - h_4$ in water level elevations,

- and inversely proportional to the length L.

From these conclusions, Henry Darcy formulated the equation, now named Darcy's equation or Darcy's law, that defines the ability of a fluid to flow through the porous material as follows

$$Q = \mathbf{K}A\frac{h_3 - h_4}{L} = \mathbf{K}A\frac{\triangle h}{L}, \tag{5.3}$$

where h is the hydraulic head in [m]. For more details about the experiment and the derivation of Darcy's law, see [134].

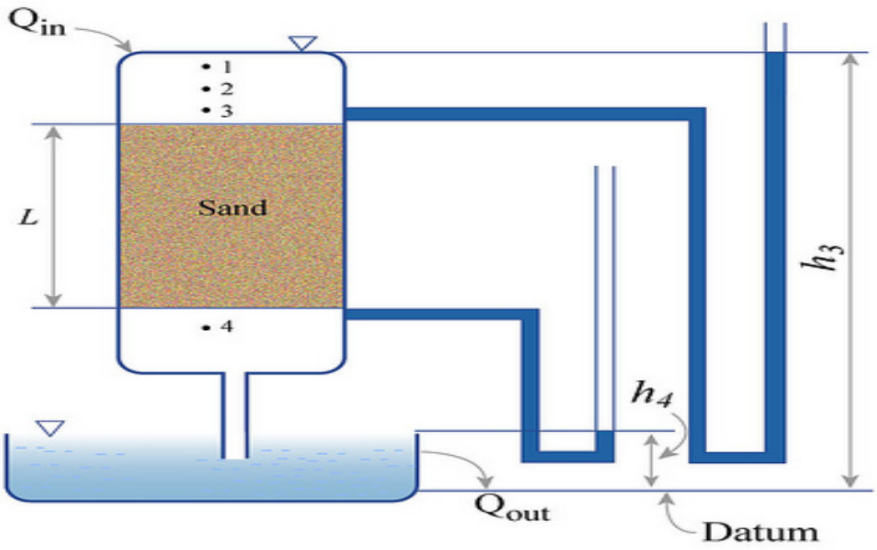


Figure 5.1: Darcy experiment [72].

The hydraulic head h is defined in [134], as the sum of the elevation and the pressure head

$$h = z + \frac{p}{\rho g}, \tag{5.4}$$

where z is the elevation, p the pressure in the fluid at the considered point. Substituting (5.4) in (5.3) yields,

$$Q = \mathbf{K}A\frac{\triangle(z + \frac{p}{\rho g})}{L}. \tag{5.5}$$

As the flow path L goes to zero, (5.5) takes the differential form

$$Q = -\mathbf{K}A\nabla(z + \frac{p}{\rho g}). \tag{5.6}$$

The minus signs on the right hand terms reflects that the hydraulic head always

decreases in the direction of flow.

Darcy's velocity and the fluid's velocity

Darcy's velocity or Darcy flux q in $[\mathrm{ms}^{-1}]$ is given by

$$q = \frac{Q}{A} = -\frac{\mathbf{k}}{\mu}(\nabla p + \rho g). \tag{5.7}$$

The Darcy flux is related to the fluid's velocity $\mathbf{v}$ and the porosity ϕ by

$$\mathbf{v} = \frac{q}{\phi} = -\frac{\mathbf{k}}{\mu\phi}(\nabla p + \rho g), \tag{5.8}$$

called the equation of motion of the fluid in the porous media. (5.8) represents the momentum balance for fluid in porous media. Given that in (5.8), we have two unknowns $\mathbf{v}$ and p, we need another equation to be able to solve the flow problem. This equation will come from mass conservation (or balance) of the fluid.

5.1.2 Mass conservation equation

In fluid mechanics, a system is defined as an identifiable quantity of matter which has a fixed volume and its mass is assumed to be constant. Therefore when we have an exchange of mass between the system and the surrounding environment, the change in the mass within the system must be equal to the difference between the incoming and the outgoing mass so as to satisfy the fundamental law of physics on the conservation of mass in the absence of source and sinks. Let us consider a fluid of density, ρ, and velocity $\mathbf{v}$, in a controlled volume V fixed in space and bounded by a smooth surface S. Let us also denote, by n, a unit vector in the direction of the outward normal to S and let M be the mass of fluid in S, see Fig 5.2.

As is shown in [203], the equation of mass conservation of a fluid is obtained as follows. The volume flow rate across S per unit area is $\mathbf{v} \cdot n$ and the mass flow rate per unit area is $\rho\mathbf{v} \cdot n$. The net rate of outflow in V is given by

$$\frac{dM}{dt} = \frac{d}{dt}\int\int\int_V \rho dV = -\int\int_S \rho\mathbf{v} \cdot n dS, \tag{5.9}$$

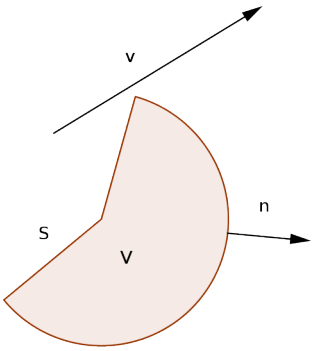


Figure 5.2: Conservation of mass in system.

where the negative sign in equation (5.9), expresses the fact that the velocity is directed out of S. According to the divergence theorem [157], the surface integral can be transformed into a volume integral as follows

$$\int\int_S \rho\mathbf{v}\cdot n dS = \int\int\int_V \nabla\cdot(\rho\mathbf{v})dV. \tag{5.10}$$

The vector $\rho\mathbf{v}$ is called the mass flux and it has the same magnitude as the mass of fluid flowing in unit time through a unit area perpendicular to the velocity vector. When we combine (5.9) and (5.10), we can write

$$\int\int\int_V \left[\frac{\partial\rho}{\partial t} + \nabla\cdot(\rho\mathbf{v})\right] dV = 0. \tag{5.11}$$

Since the choice of V is arbitrary, the integrand of equation (5.11) must be equal to zero, i.e.

$$\frac{\partial\rho}{\partial t} + \nabla\cdot(\rho\mathbf{v}) = 0 \quad \text{or} \quad \frac{\partial(\phi\rho)}{\partial t} + \nabla\cdot(\rho q) = 0. \tag{5.12}$$

This is called the mass conservation equation or continuity equation. According to [52, 51, 20], we have

$$\frac{\partial(\phi\rho)}{\partial t} = \frac{S_s}{g}\frac{\partial p}{\partial t}, \tag{5.13}$$

where S_s, called specific storage with unit $[\mathrm{m}^{-1}]$, is the volume of fluid that can be stored by compressing the porous medium and the fluid itself. By substituting (5.7)

and (5.13) in (5.12), we obtain the formulation of the mass conservation in terms of the pressure as follow

$$\frac{S_s}{g}\frac{\partial p}{\partial t} = \nabla \cdot \left(\rho \frac{\mathbf{k}}{\mu}(\nabla p + \rho g)\right). \tag{5.14}$$

In the case of an incompressible fluid, meaning the density is constant ($\rho = \rho_0$) and gravity is negligible, the mass conservation law (5.14) becomes

$$\nabla \cdot \left(\frac{\mathbf{k}}{\mu}\nabla p\right) = 0, \tag{5.15}$$

and the fluid's velocity is given by

$$\mathbf{v} = -\frac{\mathbf{k}}{\mu\phi}\nabla p. \tag{5.16}$$

Therefore to simulate the velocity of an incompressible fluid flowing through a porous medium, we first need to solve the equation of the pressure given by

$$\begin{cases} \nabla \cdot \left(-\frac{k}{\mu}\nabla p\right) & = 0, \\ p & = p_0 \quad \text{in} \quad \partial\Omega_D^1, \\ \vec{n} \cdot \left(-\frac{k}{\mu}\nabla p\right) & = p_1 \quad \text{in} \quad \partial\Omega_N^1, \end{cases} \tag{5.17}$$

where $\partial\Omega = \partial\Omega_D^1 \cup \partial\Omega_N^1$ and n is the unit outward normal vector of Ω_N^1. Finally we use (5.16), to compute the velocity field of the fluid.

5.1.3 Flow and transport by DAREs

The purpose here is to establish the conservation equation of a dissolved and chemical reactive component in porous media, which includes advection, diffusion and reaction. More details, see for example [27, 26, 186]. The fluid also called the solution, during its motion, transport a dissolved substance called solute. These substances could be toxic and possibly man-made such as contaminants transported in groundwater, or hydrocarbons transported in oil reservoirs, or CO_2 transported in transported in saline aquifers, ect. Let C denote the concentration of the solute in [kg m^{-3}].

Diffusion is a molecular mass transport process in which a solute moves from areas of higher concentration to areas of lower concentration. This phenomena can occur in the absence of velocity. The mass flux $\mathbf{J}_1$ due to the diffusion is given by Fick's law in [110]

$$\mathbf{J}_1 = -D\nabla C, \tag{5.18}$$

where D is the diffusion tensor. The negative sign before D in (5.18) indicates that the solutes move towards the area of lower concentration. In the case of an anisotropic medium, we will assume that this tensor is relative to the principal directions of the anisotropic medium. Therefore the diffusion tensor, in three dimensions, takes the form

$$D = \begin{pmatrix} D_x & 0 & 0 \\ 0 & D_y & 0 \\ 0 & 0 & D_z \end{pmatrix}. \tag{5.19}$$

Advection is the movement of a solute along with the flowing fluid in porous media. The mass flux $\mathbf{J}_2$ due to the advection is given by

$$\mathbf{J}_2 = -\mathbf{v}C, \tag{5.20}$$

for the fluid flowing at the speed $\mathbf{v}$.

The total mass flux due to the diffusion and the advection is then $\mathbf{J} = \mathbf{J}_1 + \mathbf{J}_2$. Let $R(C)$ be the reaction term that represent the variation of the concentration of the solute due to a sources or sinks. Applying the conservation of mass to the solute yields the following transport equation

$$\frac{\partial C}{\partial t} = \nabla \cdot (D\nabla C - \mathbf{v}C) + R(C). \tag{5.21}$$

For an incompressible fluid ($\nabla.\mathbf{v} = 0$), (5.21) takes the form

$$\frac{\partial C}{\partial t} = \nabla \cdot (D\nabla C) - \mathbf{v} \cdot \nabla C + R(C). \tag{5.22}$$

However, the rates of convective and diffusive transport may be quite different.

For example, the transport of pollutants in a river is dominated by convection, whereas the spreading of pollutants in a lake is dominated by diffusion. The relative strength of the diffusion and the advection can be expressed in terms of the Péclet number Pe given by

$$Pe = \frac{\mathbf{v}_0 L_0}{D_0}, \tag{5.23}$$

where $\mathbf{v}_0$ is a reference velocity, L_0 is a geometric length scale, and D_0 is a diffusion coefficient [147]. The dimensionless Peclet number then is infinite in the limit of pure convection ($D = 0$) and vanishes in the limit of pure diffusion ($\mathbf{v} = 0$).

5.2 DG method for DAREs

The purpose of this section is to review how to semidiscretize (5.22) with DG schemes and assemble the matrices, in dimension $d \geq 2$. To do so, we follow the development in Alexandre Ern and Daniele Antonio Di Pietro [80], to successively semidiscretize the diffusion equation, the advection-reaction equation, and the DAREs equation. We combine the idea presented in [80, 125] to efficiently assemble the matrices that arises from the semidiscretization.

5.2.1 Preliminaires

For a triangulation $\mathcal{T}$, the skeleton $\mathcal{F}_h$ can be decomposed into $\mathcal{F}_h^i$, the internal edges (or faces in three dimensions) and $\mathcal{F}_h^e$ the external ones. Let consider T^+, T^-, two (generic) elements sharing an edge (or face in three dimensions) $F \subset \mathcal{F}_h^i$, with respective outward normal unit vectors n^+ and n^- on F, see Fig 5.3.

Figure 5.3: **Triangulation.** We respectively show on the left and right the internal and external face with their element and outward normal vectors.

The Discontinuous Galerkin (DG) method is based on the set of discontinuous functions across the skeleton of the triangulation $\mathcal{T}$. The test space is defined as follows

$$V_h = \{v \in L^2(\Omega) : v\,|_T \in P^k(T), \forall T \in \mathcal{T}\}. \tag{5.24}$$

Then the weak formulation of (1.1), as for the FE method described in Section 2.2, can not be applied globally on Ω, but has to be split onto the each element of the triangulation. We now define some standard functions used for the application of the DG discretization. The standard notation of the **jumps function** for a function w is defined as follows

- For the internal face F i.e. $F \subset \mathcal{F}_h^i$

$$[[\nabla w_h]] = \nabla w_h^+ \cdot n^+ + \nabla w_h^- \cdot n^- = (\nabla w_h^+ - \nabla w_h^-) \cdot n^+ \tag{5.25}$$

$$[[w_h]] = w_h^+ n^+ + w_h^- n^- = (w_h^+ - w_h^-) n^+. \tag{5.26}$$

- For the external face F i.e. $F \subset \mathcal{F}_h^e$

$$[[\nabla w_h]] = \nabla w_h^+ \cdot n^+, \qquad [[w_h]] = w_h^+ n^+. \tag{5.27}$$

Note from (5.25), (5.26), (5.27) that, the jump function of a scalar-valued function is a vector-valued function and vice versa. Therefore any continuous function along the internal edges has a zero jump function. The standard notation of the **average**

function for a function w is defined as follows

- For the internal face F i.e $F \subset \mathcal{F}_h^i$

$$\{\nabla w_h\} = \frac{1}{2}(\nabla w_h^+ + \nabla w_h^-), \qquad \{w_h\} = \frac{1}{2}(w_h^+ + w_h^-). \tag{5.28}$$

- For the external face F i.e $F \subset \mathcal{F}_h^e$

$$\{\nabla w_h\} = \nabla w_h^+, \qquad \{w_h\} = w_h^+. \tag{5.29}$$

Note from (5.27) and (5.29) that for all external faces $F \subset \mathcal{F}_h^e$, we have

$$\{\nabla v_h\} \cdot [[w_h]] = [[\nabla v_h]] \{w_h\}. \tag{5.30}$$

For a real number x, we define its **positive** and **negative parts** respectively as

$$x^{\oplus} = \frac{1}{2}(\mid x \mid + x), \quad x^{\ominus} = \frac{1}{2}(\mid x \mid - x).$$

We observe that both quantities are, by definition, non negative.

5.2.2 DG for steady diffusion equations

The goal of this section is to review the IP-DG semidiscretization of the steady homogeneous and heterogeneous diffusion equations define as follow

$$\begin{cases} -\nabla \cdot (\epsilon \nabla u) = f & \text{on } \Omega \\ u = 0 & \text{on } \partial\Omega \end{cases}, \tag{5.31}$$

where $f \in L^2(\Omega)$ and $\epsilon \in \mathbb{R}^+$ are respectively the source term and the diffusion coefficient. Let us remind that we have used IP-DG method, introduced in by Wheeler [220] and Arnold [11], to investigate the one dimensional diffusion equation in Chapter 3. Here, we consider (5.31) in a dimension $d \geq 2$

Let us assume that the diffusion coefficient ϵ is constant on Ω (i.e. homogeneous). From the weak form of (5.31) at the continuous level, we design the IP-DG method

is design to approximate the solution of (5.31) in L^2-setting. In order to derive the weak form of (5.31) at the continuous level, the solution u is assumed to belong to the space $V = H_0^1(\Omega) = \{v \in H^1(\Omega) \text{ such that } v \mid_{\partial\Omega} = 0\}$. Then by the means of the integration by parts, the weak form of (5.31) is defined as follows

$$\text{Find } u \in V \text{ s.t. } b(u,w) = \int_\Omega fw, \quad \forall w \in V, \tag{5.32}$$

where the bilinear form b is given by

$$b(u,w) = \int_\Omega \epsilon \nabla u \cdot \nabla w. \tag{5.33}$$

Note that the bilinear form $b(\cdot,\cdot)$ is symmetric and bounded in $V \times V$. According to the Poincaré inequality, see [101], there is a constant C_Ω such that for all $v \in V$

$$\mid b(v,v) \mid \geq \epsilon \parallel \nabla v \parallel_{[L^2(\Omega)]^d} \geq \frac{\epsilon}{C_\Omega} \parallel v \parallel_{L^2(\Omega)} . \tag{5.34}$$

Then, owing to Theorem 2.1, the weak form (5.32) is well-posed.

Now that the weak form of (5.31) at the continuous level is reviewed, we now focus on the design of the IP-DG method to approximate the solution of (5.32) by a solution in V_h. To this end, we mimic at the discrete level the properties of the bilinear form $b(\cdot,\cdot)$ (i.e. symmetry, L^2-coercivity and boundness) that holds at the continuous level, by the means of some penalty terms. Therefore the integration by parts is operated on each element T of $\mathcal{T}$. This applied to (5.31), leads to

$$B_d(u,w_h) = \int_\Omega fw_h, \tag{5.35}$$

where $B_d(\cdot,\cdot)$ is the bilinear function defined as follows

$$B_d(u,w_h) = \sum_{T\in\mathcal{T}} \int_T \epsilon \nabla_h u \cdot \nabla_h w_h - \sum_{T\in\mathcal{T}} \int_{\partial T} (\vec{n} \cdot \epsilon \nabla_h u) w_h, \tag{5.36}$$

for all $w_h \in V_h$. Using the definition of the jump function in (5.25)- (5.27), we can transform the second term in the right hand side of (5.36) into a sum of integrals

over the the faces as follows

$$\sum_{T\in\mathcal{T}}\int_{\partial T}(\vec{n}\cdot\epsilon\nabla_h u)w_h=\sum_{F\in\mathcal{F}_h}\int_F[[(\epsilon\nabla_h u)w_h]]. \tag{5.37}$$

As for the 1D case in Chapter 3, by using (3.27) with the definition of the jump and the average functions in (5.25)-(5.29), we obtain

$$[[\epsilon\nabla_h u w_h]]=\begin{cases}\{\epsilon\nabla_h u\}\cdot[[w_h]]+[[\epsilon\nabla_h u]]\{w_h\}, & \forall F\in\mathcal{F}_h^i\\ \{\epsilon\nabla_h u\}\cdot[[w_h]], & \forall F\in\mathcal{F}_h^e\end{cases}. \tag{5.38}$$

Since ϵ is constant, then according to [80], by assuming that the exact solution $u\in V\cap W^{2,1}(\Omega)$, we have

$$\forall F\in\mathcal{F}_h,\ [[u]]=0 \quad \text{and} \quad \forall F\in\mathcal{F}_h^i,\ [[\epsilon\nabla u]]=0. \tag{5.39}$$

Therefore, combining (5.38) and (5.39) in (5.37) yields

$$\sum_{T\in\mathcal{T}}\int_{\partial T}(\vec{n}\cdot\epsilon\nabla_h u)w_h=\sum_{F\in\mathcal{F}_h}\int_F\{\epsilon\nabla_h u\}\cdot[[w_h]],$$

and then the bilinear form $B_d(\cdot,\cdot)$ becomes

$$B_d(u,w_h)=\sum_{T\in\mathcal{T}}\int_T\epsilon\nabla_h u\cdot\nabla_h w_h-\sum_{F\in\mathcal{F}_h}\int_F\{\epsilon\nabla_h u\}\cdot[[w_h]]. \tag{5.40}$$

Note from (5.40) that the bilinear form $B_d(\cdot,\cdot)$ is nonsymmetric, owing to the second term on the right hand side of (5.40). Since a desirable property of the discrete bilinear form is to preserve the original symmetry of the bilinear form $b(\cdot,\cdot)$, we consider the bilinear form

$$B_d^s(v,w_h)=\sum_{T\in\mathcal{T}}\int_T\epsilon\nabla_h v\cdot\nabla_h w_h-\sum_{F\in\mathcal{F}_h}\int_F\left(\{\epsilon\nabla_h v\}\cdot[[w_h]]+\{\epsilon\nabla_h w_h\}\cdot[[v]]\right), \tag{5.41}$$

for all $v,w_h\in V_h$. Indeed, symmetry can simplify the resolution process of the resulting linear system and furthermore, it is a natural ingredient to derive optimal

L^2-norm error estimation [11, 80]. The second term on the right-hand side of (5.41) is called the symmetric term.

Other desirable properties are the discrete L^2-coercivity and boundness on the broken polynomial space V_h with respect to a suitable norm. Indeed, these properties will ensure the well posedness of the discrete weak form, owing to Theorem 2.1. But the difficulty with the discrete bilinear form $B_d^s(\cdot,\cdot)$ defined by (5.41) is that, for all $w_h \in V_h$

$$B_d^s(w_h, w_h) = \int_\Omega \epsilon \parallel \nabla_h w_h \parallel_{\mathbb{R}^d}^2 -2 \sum_{F\in\mathcal{F}_h} \int_F \{\epsilon \nabla_h w_h\} \cdot [[w_h]],$$

and the second term on the right-hand side has no a priori sign so that, without adding a further term, there is no hope for discrete coercivity. To achieve discrete coercivity, we add to $B_d^s(\cdot,\cdot)$, defined by (5.41), a term penalizing interface and boundary jumps. Namely we set as in [11, 80]

$$B_d^{sip}(u_h, w_h) = B_d^s(u_h, w_h) + \overbrace{\sum_{F\in\mathcal{F}_h} \frac{\eta\epsilon}{h_F} \int_F [[u_h]] \cdot [[w_h]]}^{S_h(u_h,w_h)}, \tag{5.42}$$

where h_F is a local length scale associated with the face $F \in \mathcal{F}_h$, the quantity $\eta > 0$ denotes a user-dependent parameter which is independent of the diffusion coefficient. The local length scale h_F is set to the diameter of the face F in the dimension $d \geq 2$. If the penalty parameter η is large enough then the bilinear form $B_d^{sip}(\cdot,\cdot)$ is coercive in V_h, see [11, 80]. The bilinear form $B_d^{sip}(\cdot,\cdot)$ is called the SIPG bilinear form. Note from (5.42) that the discrete bilinear form $B_d^{sip}(\cdot,\cdot)$ is still symmetric since the bilinear forms $B_d^s(\cdot,\cdot)$ and the so-called penalty bilinear form S_h are symmetric. Due to (5.39) for all $w_h \in V_h$ and the exact solution u, we have $B_d^{sip}(u, w_h) = B_d(u, w_h)$. This shows that the consistency of B_d has not changed with the terms added. Therefore, owing to Theorem 2.1, a well-posed weak form of (5.31) can be formulated as follows

$$\text{Find } u_h \in V_h \text{ s.t. } B_d^{sip}(u_h, w_h) = \int_\Omega f w_h, \quad \forall w_h \in V_h. \tag{5.43}$$

Now that the SIPG weak form has been presented for the steady diffusion equa-

tion subject to the nonhomogeneous Dirichlet boundary condition, we focus on the formulation of the SIPG weak form in the case of homogeneous Dirichlet, Neumann and Robin boundary condition. The importance of this investigation is to be able to weakly enforce any type of boundary condition.

- Firstly, let us consider the steady diffusion equation subject to nonhomogeneous Dirichlet boundary condition i.e. $u = g \quad \text{on } \partial\Omega$, with $g \in L^2(\partial\Omega)$. In this case the jump function of u is no longer equal to zero across the external face $F \in \mathcal{F}_h^e$, as in (5.39). Then the weak form of (5.31) can be formulated as follows : find u_h in V_h such that

$$B_d^{sip}(u_h, w_h) = \int_\Omega f w_h - \sum_{F \in \mathcal{F}_h^e} \int_F (\epsilon \nabla_h w_h) g + \sum_{F \in \mathcal{F}_h^e} \frac{\eta \epsilon}{h_F} \int_F g w_h, \tag{5.44}$$

  for all w_h in V_h. The second and third terms on the right-hand side of (5.44) appeared while adding the terms to mimic at the discrete level the symmetry and the coercivity of the continuous weak form.

- Secondly, let us consider the steady diffusion equation subject to homogeneous Neumann boundary condition i.e. $n \cdot \nabla u = 0 \quad \text{on } \partial\Omega$. In this case the jump function of the diffusive flux is equal to zero across the external face $F \in \mathcal{F}_h^e$. Then (5.40) can be reduced as follows

$$B_d(u_h, w_h) = \sum_{T \in \mathcal{T}} \int_T \epsilon \nabla_h u_h \nabla_h w_h - \sum_{F \in \mathcal{F}_h^i} \int_F \{\epsilon \nabla_h u\} \cdot [[w_h]]. \tag{5.45}$$

  By adding the terms to mimic at the discrete level the symmetry and the coercivity of the continuous weak form, we can formulate the weak form of (5.31) as follows : find u_h in V_h such that

$$A(u_h, w_h) = \int_\Omega f w_h \ \ \forall w_h \in V_h,$$

where the bilinear term A is defined by

$$A(u_h, w_h) = \sum_{T\in\mathcal{T}} \int_T \epsilon \nabla_h u_h \cdot \nabla_h w_h - \sum_{F\in\mathcal{F}_h^i} \int_F \{\epsilon \nabla_h u_h\} \cdot [[w_h]] - \sum_{F\in\mathcal{F}_h^i} \int_F \{\epsilon \nabla_h w_h\} \cdot [[u_h]] + \sum_{F\in\mathcal{F}_h^i} \frac{\eta\epsilon}{h_F} \int_F [[u_h]] \cdot [[w_h]].$$

- Finally, let us consider the steady diffusion equation subject to the Robin boundary condition $\lambda u + n \cdot \epsilon \nabla u = g \quad$ on $\partial\Omega$, with $g \in L^2(\partial\Omega),\ \lambda \in L^\infty(\partial\Omega)$ and λ non negative almost everywhere on $\partial\Omega$. Then the weak form of (5.31) can be formulated as follows : find u_h in V_h such that

$$A(u_h, w_h) + \sum_{F\in\mathcal{F}_h^e} \int_F \lambda u_h w_h = (f, w_h) + \sum_{F\in\mathcal{F}_h^e} \int_F g w_h \ \ \forall w_h \in V_h.$$

  To derive the weak form in this case, we estimate the jump function of the diffusive flux (i.e. $n \cdot \nabla u$) from the boundary condition, then we substitute the result in (5.40) and follow the process to mimic at the discrete level the symmetry and the coercivity of the continuous weak form.

We examine the convergence of the IP-DG method reviewed here for an unsteady and homogeneous diffusion equation in Section 5.3.2.

Theorem 5.1 (IP-DG for heterogeneous diffusion equation)**.** *Let assume that there is a partition of Ω in a set $P_\Omega = \{\Omega_i, i = 1, \cdots, n\}$ of polyhedrons such that the restriction of ϵ to each polyhedron Ω_i is constant. If the mesh $\mathcal{T}_h$ is such that each element $T \in \mathcal{T}_h$ belongs to only one polyhedron Ω_i then the weak form of* (5.31) *takes the form*

$$\text{Find } u_h \in V_h \text{ s.t. } B_d^{swip}(u_h, w_h) = \int_\Omega f w_h, \ \ \forall w_h \in V_h, \tag{5.46}$$

where the bilinear form B_d^{swip} is given by

$$B_d^{swip}(u_h, w_h) = \sum_{T\in\mathcal{T}} \int_T \epsilon \nabla_h u_h \cdot \nabla_h w_h - \sum_{F\in\mathcal{F}_h} \int_F \{\epsilon \nabla_h u_h\}_w^F \cdot [[w_h]] - \sum_{F\in\mathcal{F}_h} \int_F \{\epsilon \nabla_h w_h\}_w^F \cdot [[u_h]] + \sum_{F\in\mathcal{F}_h} \frac{\eta\gamma_F^\epsilon}{h_F} \int_F [[u_h]] \cdot [[w_h]].$$

Here, the weighted average $\{\cdot\}_w^F$ *and the diffusion dependent penalty parameter* γ_F^ϵ *are define as follow*

$$\{v\}_w^F = \begin{cases} \dfrac{\epsilon_2 v_1 + \epsilon_1 v_2}{\epsilon_1 + \epsilon_2} & on\ F = \partial T_1 \cap \partial T_2 \in \mathcal{F}_h^i \\ v_1 & on\ F = \partial T_1 \cap \partial\Omega \in \mathcal{F}_h^e \end{cases},$$

$$\gamma_F^\epsilon = \begin{cases} \dfrac{2\epsilon_1\epsilon_2}{\epsilon_1 + \epsilon_2} & on\ F = \partial T_1 \cap \partial T_2 \in \mathcal{F}_h^i \\ \epsilon_1 & on\ F = \partial T_1 \cap \partial\Omega \in \mathcal{F}_h^e \end{cases},$$

where $\epsilon_i = \epsilon\,|_{T_i}$ *and* $v_i = v\,|_{T_i}$ *for all* $i \in \{1, 2\}$.

The bilinear form B_d^{swip}, called the symmetric weighted interior penalty galerkin (SWIPG) method, was introduced in 2003 by Dryja [87] and further analysed by Ern et al. [81, 98]. Note that if the diffusion coefficient is constant on Ω, the bilinear forms B_d^{swip} and B_d^{sip} are equal. In Section 5.3.5, we use B_d^{swip} to simulate the velocity of the fluid trough a medium with fracture.

5.2.3 DG for steady advection-reaction equations

Let us consider the steady advection-reaction equation with homogeneous inflow boundary condition

$$\begin{cases} \beta \cdot \nabla u + \mu u = f & \text{on } \Omega \\ u = 0 & \text{on } \partial\Omega^- \end{cases}. \tag{5.47}$$

This is one of the simplest model problems based on a linear, scalar, steady first-order PDE. The unknown function u is scalar-valued and represents, e.g., a solute concentration; β is the $\mathbb{R}^d$-valued advective velocity, μ the reaction coefficient, f the source term, and $\partial\Omega^-$ denotes the inflow part of the boundary of Ω, namely

$$\partial\Omega^- = \{x \in \partial\Omega \mid \beta(x) \cdot \vec{n}(x) < 0\}. \tag{5.48}$$

The main goal of this section is to review the analysis of the DG method described in [80], to approximate the solution of (5.47). Let us recall that the application of the DG method to unsteady advection and reaction has been propelled by the innovative works of Cockburn et al [62, 61, 64].

Since the DG methods are essentially tailored to approximate PDEs in an L^2-setting where discrete stability is enhanced by suitable least-squares penalties, the most natural weak formulation at the continuous level is that based on the concept of graph space. So, in order to design the DG method for (5.47), we first examine the weak formulation of (5.47) at the continuous level. For well posedness of the weak formulation of (5.47), we assume that $\mu \in L^{\infty}(\Omega)$, $\beta \in [Lip(\Omega)]^d$, $f \in L^2(\Omega)$ and there is a real number $\mu_0 > 0$ such that

$$\mu - \frac{1}{2}\nabla \cdot \beta \geq \mu_0 \quad a.e. \text{ in } \Omega. \tag{5.49}$$

The regularity on β can be weakened at least to $\| \beta \|_{[L^{\infty}(\Omega)]^d}$ and $\| \nabla \cdot \beta \|_{L^{\infty}(\Omega)}$ bounded. For more details about the assumptions on the data and the formulation of the weak form, see for example [80]. Before we consider the weak formulation of (5.47), let us first define the graph space and the trace operator. They respectively specify the functional space in which the solution of (5.47) is sought and the mathematical meaning of the boundary condition.

Definition 5.1. The graph space is a Hilbert space given by $V = \{v \in L^2(\Omega) \mid \beta \cdot \nabla v \in L^2(\Omega)\}$ and equipped with the scalar product

$$(u, v)_V = (u, v)_{L^2(\Omega)} + (\beta \cdot \nabla u, \beta \cdot \nabla v)_{L^2(\Omega)}.$$

Let the set $L^2(| \beta \cdot n |; \partial\Omega)$ be defined as follows

$$L^2(| \beta \cdot n |; \partial\Omega) = \{v \in L^1(\partial\Omega) \text{ such that } \int_{\partial\Omega} | \beta \cdot n | v^2 < \infty\}.$$

Recalling the definition of the inflow boundary with (5.48), we also define the outflow boundary as

$$\partial\Omega^+ = \{x \in \partial\Omega \mid \beta(x) \cdot \vec{n}(x) > 0\}.$$

We assume that the inflow and outflow boundaries are well separated (i.e. their intersection is empty). The following result is very important since it allows us to define the trace operator on the graph space and to use an integration by parts

formula.

Lemma 5.1. [80] The trace operator

$$\gamma : C^{\infty}(\bar{\Omega}) \ni v \longmapsto \gamma(v) := v \mid_{\partial\Omega} \in L^2(\mid \beta \cdot n \mid; \partial\Omega),$$

can be extended to V, meaning that there is a constant C_γ such that, for all $v \in V$,

$$\parallel \gamma(v) \parallel_{L^2(|\beta\cdot n|;\partial\Omega)} \leq C_\gamma \parallel v \parallel_V .$$

Moreover, the following integration by parts formula holds for all v, w in V

$$\int_\Omega \nabla \cdot (\beta v) w + (\beta \cdot \nabla w) v = \int_{\partial\Omega} (\beta \cdot n) \gamma(v) \gamma(w). \tag{5.50}$$

We now focus on the derivation of the weak form of (5.47). Using the graph space V, (5.47) can be cast into the weak form at the continuous level as follows

$$\text{Find } u \in V \quad \text{such that } \; a(u, w) = \int_\Omega f w \quad \text{for all } w \in V, \tag{5.51}$$

where $a(\cdot, \cdot)$ is a bounded bilinear form in $V \times V$ given by

$$a(v, w) = \int_\Omega \mu v w + \int_\Omega (\beta \cdot \nabla v) w + \int_{\partial\Omega} (\beta \cdot n)^{\ominus} v w. \tag{5.52}$$

It has been shown, in [80], that (5.51) is well-posed. The following result tells us in which sense the solution of (5.51) solves (5.47), specifically it shows that the boundary condition is weakly enforced in (5.51).

Lemma 5.2. [80] If the function u is a solution of (5.51), then we have

$$\begin{cases} \beta \cdot \nabla u + \mu u = f & \text{a.e. in } \Omega \\ u = 0 & \text{a.e. in } \partial\Omega^- \end{cases} . \tag{5.53}$$

Now that (5.51) (i.e. the weak form of (5.47) at the continuous level) has been proposed, we now focus on the design and the analysis of the DG method to approximate the solution of (5.51). To this end, we assume that the exact solution

u, belongs to $V_* = V \cap V_h$, where V_h is given by (5.24). This assumption implies that for all elements T of a triangulation $\mathcal{T}_h$, the restriction $u\,|_T$ has traces almost everywhere on each face $F \in \mathcal{F}_T$, and these traces belong to $L^2(F)$ [80].

Lemma 5.3. [80] The exact solution $u \in V_h$ is such that, for all $F \in \mathcal{F}_h^i$,

$$\beta \cdot [[u]](x) = 0 \quad \text{a.e. } x \in F. \tag{5.54}$$

The starting point in the process of defining the suitable DG method here is to introduce a discrete bilinear form $a_h^0(\cdot,\cdot)$ simply derived from the exact bilinear form $a(\cdot,\cdot)$, by replacing the advective derivative $\beta \cdot \nabla$ by its broken counterpart $\beta \cdot \nabla_h$ defined as follows

$$(\beta \cdot \nabla_h v_h)\,|_T = \beta \cdot \nabla(v_h\,|_T),$$

for all $v_h \in V_v$ and $T \in \mathcal{T}_h$. We then have for all $v_h, w_h \in V_h$

$$a_h^0(v_h, w_h) = \int_\Omega \mu v_h w_h + \int_\Omega (\beta \cdot \nabla_h v_h) w_h + \int_\Omega (\beta \cdot n)^{\ominus} v_h w_h. \tag{5.55}$$

Let us now examine the discrete coercivity of the bilinear form $a_h^0(\cdot,\cdot)$. By definition, we have for all $v_h \in V_h$

$$a_h^0(v_h, v_h) = \int_\Omega \mu v_h^2 + \sum_{T \in \mathcal{T}_h} \int_T (\beta \cdot \nabla_h v_h) v_h + \int_{\partial\Omega} (\beta \cdot n)^{\ominus} v_h^2.$$

Applying integration by parts on each mesh element T, the above equation becomes for all $v_h \in V_h$

$$a_h^0(v_h, v_h) = \int_\Omega (\mu - \frac{1}{2}\nabla \cdot \beta) v_h^2 + \sum_{T \in \mathcal{T}_h} \int_{\partial T} \frac{1}{2} \beta \cdot n v_h^2 + \int_{\partial\Omega} (\beta \cdot n)^{\ominus} v_h^2. \tag{5.56}$$

According to the definition of jump functions and the continuity of the function β across the interfaces, the second term on the right hand side of (5.56) can be reformulated as the sum over the faces, namely

$$\sum_{T \in \mathcal{T}_h} \int_{\partial T} \frac{1}{2} \beta \cdot n v_h^2 = \sum_{F \in \mathcal{F}_h^i} \int_F \frac{1}{2} \beta \cdot [[v_h^2]] + \sum_{F \in \mathcal{F}_h^b} \int_F \frac{1}{2} (\beta \cdot n) v_h^2. \tag{5.57}$$

For all $F \in \mathcal{F}_h^i$, there exist two elements T_1, T_2 such that $F = \partial T_1 \cap T_2$. Let us introduce the notation $v_i = v_h \mid_{T_i},\ i \in \{1,2\}$ and n_1 the outward normal to T_1 on F, then we have

$$\frac{1}{2}[[v_h^2]] = \frac{1}{2}(v_1^2 - v_2^2)n_1 = \frac{1}{2}(v_1 + v_2)(v_1 - v_2)n_1 = \{v_h\}[[v_h]]. \tag{5.58}$$

Substituting (5.57) and (5.58) in (5.56), yields for all $v_h \in V_h$ and $\gamma = (\mu - \frac{1}{2}\nabla \cdot \beta)$

$$\begin{aligned} a_h^0(v_h, v_h) &= \int_\Omega \gamma v_h^2 + \sum_{F \in \mathcal{F}_h^i} \int_F \beta \cdot [[v_h]]\{v_h\} + \sum_{F \in \mathcal{F}_h^b} \int_F \frac{1}{2}(\beta \cdot n) v_h^2 + \int_{\partial\Omega} (\beta \cdot n)^{\ominus} v_h^2 \\ &= \int_\Omega \gamma v_h^2 + \sum_{F \in \mathcal{F}_h^i} \int_F \beta \cdot [[v_h]]\{v_h\} + \int_{\partial\Omega} \frac{1}{2}(\beta \cdot n) v_h^2 + \int_{\partial\Omega} (\beta \cdot n)^{\ominus} v_h^2 \\ &= \int_\Omega \gamma v_h^2 + \sum_{F \in \mathcal{F}_h^i} \int_F \beta \cdot [[v_h]]\{v_h\} + \int_{\partial\Omega} \mid \beta \cdot n \mid v_h^2. \end{aligned} \tag{5.59}$$

The second term on the right-hand side, involving interfaces, has no sign a priori. Therefore, it must be removed in order to ensure the discrete coercivity. This can be achieved if we set, for all $v_h, w_h \in V_h$

$$a_h^1(v_h, w_h) = a_h^0(v_h, w_h) - \sum_{F \in \mathcal{F}_h^i} \int_F \beta \cdot [[v_h]]\{w_h\},$$

since (5.54) holds. To strengthen the discrete coercivity, it is proposed in [80] to add the stabilisation bilinear form

$$S_h(v_h, w_h) = \sum_{F \in \mathcal{F}_h^i} \int_F \frac{\eta}{2} \mid \beta \cdot n \mid [[v_h]] \cdot [[w_h]], \tag{5.60}$$

where $\eta > 0$ is a user dependent parameter. We then consider the bilinear form B_a^{up}, so called upwind DG bilinear form, defined as follows

$$\begin{aligned} B_a^{up}(v_h, w_h) &= a_h^1(v_h, w_h) + S_h(v_h, w_h) \\ &= \int_\Omega (\mu v_h w_h + (\beta \cdot \nabla_h v_h) w_h) + \int_{\partial\Omega} (\beta \cdot n)^{\ominus} v_h w_h \\ &\quad - \sum_{F \in \mathcal{F}_h^i} \left(\int_F \beta \cdot [[v_h]]\{w_h\} - \int_F \frac{\eta}{2} \mid \beta \cdot n \mid [[v_h]] \cdot [[w_h]] \right), \end{aligned} \tag{5.61}$$

for all $v_h, w_h \in V_h$. Therefore the weak form of the equation (5.47) at the discontinuous level is defined as follows

$$\text{Find } u_h \in V_h \text{ such that } B_a^{up}(u_h, w_h) = \int_\Omega f w_h, \ \ \forall w_h \in V_h. \tag{5.62}$$

For more details on the well-posedness and the error estimation of (5.62), see [80]. We now examine how to weakly enforce the nonhomogeneous inflow boundary condition.

Lemma 5.4. [80] In the case of the steady advection-reaction equation with nonhomogeneous inflow boundary condition i.e.

$$\begin{cases} \beta \cdot \nabla u + \mu u = f & \text{on } \Omega \\ u = g & \text{on } \partial\Omega^- \end{cases}, \tag{5.63}$$

the weak formulation is defined as follows

$$\text{Find } u_h \in V_h \text{ s.t. } B_a^{up}(u_h, w_h) = (f, w_h) + \int_{\partial\Omega} (\beta \cdot n)^{\ominus} g w_h, \text{ for all } w_h \in V_h.$$

To validate the DG method proposed here for (5.47), We examine the convergence in space for a given β, μ and f in Section 5.3.1.

5.2.4 DG for DAREs equations and Discretization

The purpose of this section is to present the DG weak form of (1.1) by combining the upwind and the SIPG method (respectively to handle advection-reaction terms and diffusion terms). Once the weak form of the DAREs is presented, we investigate the semi discretization of the weak form obtained, which leads to a system of ODEs.

Let us first consider the steady advection-diffusion-reaction equation with homogeneous boundary condition as follows

$$\begin{cases} -\nabla(\epsilon \nabla u) + \beta \cdot \nabla u + \mu u = f & \text{on } \Omega \\ u = 0 & \text{on } \partial\Omega \end{cases}. \tag{5.64}$$

According to the analysis presented in Section 5.2.3 and Section 5.2.2, the well-posed DG weak form of (5.64) can be formulated as

$$\text{Find } u_h \in V_h \text{ s.t. } B_d^{sip}(u_h, w_h) + B_a^{up}(u_h, w_h) = \int_\Omega f w_h, \quad \forall w_h \in V_h, \tag{5.65}$$

where the bilinear form B_a^{up} and B_d^{sip} are respectively given by (5.61) and (5.42). For the full details on the proof of the well posedness of (5.65), see for example [11, 80]. Note that if $\mu - \frac{1}{2}\nabla \cdot \beta = 0$, the bilinear form $B_a^{up} + B_d^{sip}$ still coecirve.

More generally, for unsteady advection-diffusion-reaction subject to mixed boundary condition (Dirichlet, Neumann or Robin conditions) and some initial condition ($u(\cdot, t = t_0) = u_0$), the weak form takes the form : For all $t \in [t_0, t_f]$, find $u_h \in V_h$ such that

$$\int_\Omega \frac{\partial u_h}{\partial t} w_h + \mathcal{S}(u_h, w_h) = \int_\Omega f w_h + \mathcal{B}(w_h), \forall w_h \in V_h \tag{5.66}$$

where the bilinear form $\mathcal{S}$ (that handle the advection and diffusion terms) and the linear form $\mathcal{B}$ (that handle the boundary condition) can be derived with the upwind and SIPG method. In this case the bilinear form $\mathcal{S}$ and the linear form $\mathcal{B}$ can be formulated as follows

$$\mathcal{S}(u_h, w_h) = \sum_{T \in \mathcal{T}} \underbrace{\int_T \{\cdots\}}_{\mathcal{S}_T(u_h, w_h)} + \sum_{F \in \mathcal{F}_h^i} \underbrace{\int_F \{\cdots\}}_{\mathcal{S}_F^i(u_h, w_h)} + \sum_{F \in \mathcal{F}_h^e} \underbrace{\int_F \{\cdots\}}_{\mathcal{S}_F^e(u_h, w_h)}, \tag{5.67}$$

$$\mathcal{B}(w_h) = \sum_{F \in \mathcal{F}_h^e} \underbrace{\int_F \{\cdots\}}_{\mathcal{B}_F^e(w_h)}. \tag{5.68}$$

Now that the DG weak form has been presented, we focus on its discretization. To do so, we consider a finite dimensional approximation of the space V_h at the discrete level. Due to the fact that the restriction of a function $u_h \in V_h$ to a given element $T \in \mathcal{T}$ can be chosen independently of its restriction to other elements, we reduced the support of the basis functions to a single element. As consequence, the communication between the mesh elements is reduced. For a structured mesh, it is natural to assume that the global enumeration of the degrees of freedom is such that the local degrees of freedom are numbered contiguously for each mesh element.

Therefore the approximation of the global space V_h is given by

$$V_h = \oplus_{j=1}^{N_\mathcal{T}} V_h^j, \quad V_h^j = \text{ span}\left\{\phi_i^{T_j},\ i \in D_{T_j}\right\}, \tag{5.69}$$

where V_h^j is the the locally defined spaces, $N_\mathcal{T}$ is the total number of the element $T_j \in \mathcal{T}$, the set $D_{T_j} = \{1, \cdots, N_{dof}^{T_j}\}$ collects the the local indices of the $N_{dof}^{T_j}$ degrees of freedom for the mesh element $T_j \in \mathcal{T}$ and where

$$\text{supp}(\phi_i^{T_j}) = \bar{T_j}, \quad \forall T_j \in \mathcal{T}_h,\ \forall i \in D_{T_j}. \tag{5.70}$$

The dimension of V_h is then equal to $\mathcal{N} = \sum_{j=1}^{N_\mathcal{T}} N_{dof}^{T_j}$. The number of degrees of freedom $N_{dof}^{T_j}$ can vary from element to element. Different choices for the local basis $\left\{\phi_i^{T_j},\ i \in D_{T_j}\right\}$ are investigated in [125, 80]. The global solution $u(t,p)$, for a given point $p \in \mathbb{R}^d$ and time t, is then assumed to be approximated as follows

$$u(t,p) \approx u_h(t,p) = \sum_{j=1}^{N_\mathcal{T}} \sum_{i=1}^{N_{dof}^{T_j}} X_i^{T_j}(t)\phi_i^{T_j}(p). \tag{5.71}$$

For the steady case, $X_i^{T_j}$ are constant independent of the time. In the limit of $(N_\mathcal{T}, N_{dof}^{T_j}) \longrightarrow \infty$, we assume that u_h converges uniformly to the global solution u. Therefore, by substituting (5.71) in (5.66) and considering the function w_h as the basis functions $\phi_i^{T_j}$, we obtain a system of ODEs of unknown X defined as follows

$$\mathbb{M}\frac{d}{dt}X + \mathbb{S}X = \mathbb{F} + \mathbb{B}, \tag{5.72}$$

where $X,\ \mathbb{F},\ \mathbb{B} \in \mathbb{R}^\mathcal{N}$ and matrices $\mathbb{M},\ \mathbb{S} \in \mathbb{R}^{\mathcal{N}\times\mathcal{N}}$ are given by

$$X = \left[X_i^{T_j}\right]_{i\in D_{T_j}}^{T_j\in\mathcal{T}},\ \mathbb{B} = \left[\mathcal{B}(\phi_i^{T_j})\right]_{i\in D_{T_j}}^{T_j\in\mathcal{T}},\ \mathbb{F} = \left[\int_\Omega f\phi_i^{T_j}\right]_{i\in D_{T_j}}^{T_j\in\mathcal{T}}, \tag{5.73}$$

$$\mathbb{S} = \left[\mathcal{S}(\phi_i^{T_j}, \phi_n^{T_m})\right]_{i\in D_{T_j}, n\in D_{T_n}}^{T_j, T_m\in\mathcal{T}},\ \mathbb{M} = \left[\int_\Omega \phi_i^{T_j}\phi_n^{T_m}\right]_{i\in D_{T_j}, n\in D_{T_m}}^{T_j, T_m\in\mathcal{T}}. \tag{5.74}$$

As in Chapter 3, the matrices $\mathbb{M},\ \mathbb{S}$ are respectively called the mass and stiffness matrices. Once the vectors $\mathbb{F},\ \mathbb{B}$ and the matrices $\mathbb{M},\ \mathbb{S}$ are assembled, (5.72) can

be solved for X by means of the standard time discretization methods presented in Section 2.3.

5.2.5 Implementation of DG method for DAREs

In this section, we present an efficient way to assemble the matrices $\mathbb{M}$, $\mathbb{S}$ and the vectors $\mathbb{F}$, $\mathbb{B}$. The methodology used to assemble the matrix in the case of DG differs from the one used for FE methods because the degrees of freedom associated with each mesh element are decoupled from those associated with the remaining elements; and the terms involving integrals on interfaces are generally present. The simplest way to assemble these matrices and vectors is to generate the basis function and compute every single entry. But due to the local construction of the basis functions i.e (5.70), most of the entries are equal to zero and even more most of the terms in their formula is null. So the efficient way to assemble these entities is first to identify the non zeros entries and then the non zero terms in their formula.

Firstly, let us consider the assembling of the mass matrix. According to the local definition of the basis functions to single mesh elements, the mass matrix $\mathbb{M}$ can be block-partitioned as follows

$$\mathbb{M} = \begin{pmatrix} \mathbb{M}^{T_1T_1} & 0 & \cdots & 0 \\ 0 & \mathbb{M}^{T_2T_2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \mathbb{M}^{T_{N_T}T_{N_T}} \end{pmatrix}, \tag{5.75}$$

where for all $T_j \in \mathcal{T}$, the block matrix $\mathbb{M}^{T_jT_j} \in \mathbb{R}^{N_{dof}^{T_j} \times N_{dof}^{T_j}}$ is the local mass matrix corresponding to the element $T_j \in \mathcal{T}$ and is given by

$$\mathbb{M}^{T_jT_j} = \left[\int_{T_j} \phi_i^{T_j} \phi_n^{T_j} \right]_{i \in D_{T_j}, n \in D_{T_j}}. \tag{5.76}$$

Note from (5.76) that the local mass matrices are symmetric definite positive. Thus the mass matrix is easily invertible due to its block diagonal structure. A typical situation where this inverse is needed is when solving (5.72) with ETD, EXPR or

ROCK2 time solver, see Section 2.3.

Secondly, let us consider the assembling of the stiffness matrix $\mathbb{S}$. In general, the global stiffness matrix can be block-partitioned as follows

$$\mathbb{S} = \begin{pmatrix} \mathbb{S}^{T_1T_1} & \mathbb{S}^{T_1T_2} & \dots & \mathbb{S}^{T_1T_{N_T}} \\ \mathbb{S}^{T_2T_1} & \mathbb{S}^{T_2T_2} & \dots & \mathbb{S}^{T_2T_{N_T}} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbb{S}^{T_{N_T}T_1} & \mathbb{S}^{T_{N_T}T_2} & \dots & \mathbb{S}^{T_{N_T}T_{N_T}} \end{pmatrix}, \tag{5.77}$$

where for all $T_j, T_m \in \mathcal{T}$, the block matrix $\mathbb{S}^{T_jT_m} \in \mathbb{R}^{N_{dof}^{T_j} \times N_{dof}^{T_j}}$ is given by

$$\mathbb{S}^{T_jT_m} = \left[\mathcal{S}(\phi_n^{T_m}, \phi_i^{T_j})\right]_{i \in D_{T_j}, n \in D_{T_m}}. \tag{5.78}$$

Combining the definition of the bilinear form $\mathcal{S}$ and the localization of basis functions to single mesh elements i.e. (5.70), we can conclude that the block matrix $\mathbb{S}^{T_jT_m}$ is non zero only if $T_j = T_m$ (diagonal block) or if T_j and T_m share a common interface $F \in \mathcal{F}_h$. Each summation of the bilinear function $\mathcal{S}$ in (5.67) yields a loop over the corresponding mesh entities to assemble local contributions into the global stiffness matrix. Moreover, owing to (5.70), for a given $T \in \mathcal{T},\ F_1 \in \mathcal{F}_h^i,\ F_2 \in \mathcal{F}_h^e$, we have

$$\mathcal{S}_T(\phi_n^{T_m}, \phi_i^{T_j}) \neq 0 \Longleftrightarrow T = T_m = T_j, \tag{5.79}$$

$$\mathcal{S}_{F_1}^i(\phi_n^{T_m}, \phi_i^{T_j}) \neq 0 \Longleftrightarrow F_1 \subseteq \partial T_m \cap \partial T_j, \tag{5.80}$$

$$\mathcal{S}_{F_2}^e(\phi_n^{T_m}, \phi_i^{T_j}) \neq 0 \Longleftrightarrow F_2 = \partial T \cap \partial\Omega \text{ and } T = T_m = T_j. \tag{5.81}$$

From (5.79), we can say that the local stiffness matrix stemming from the volume contribution of a generic mesh element $T_j \in \mathcal{T}$ is, the block matrix $\mathbb{S}^{T_j} \in \mathbb{R}^{N_{dof}^{T_j} \times N_{dof}^{T_j}}$, given by

$$\mathbb{S}^{T_j} = \left[\mathcal{S}_{T_j}(\phi_n^{T_j}, \phi_i^{T_j})\right]_{i,n \in D_{T_j}}, \tag{5.82}$$

and it contributes to the diagonal block $\mathbb{S}^{T_jT_j}$ of the global stiffness matrix $\mathbb{S}$. From (5.80), we can conclude that an interface $F \in \mathcal{F}_h^i$ (i.e. there exist T_j and T_m with $j < m$ such that $F = \partial T_j \cap \partial T_m$) contributes to four blocks of the global stiffness

matrix $\mathbb{S}$ and the local stiffness matrix stemming from the interface contribution can be block-partitioned in the form

$$\mathbb{S}_F^i = \begin{pmatrix} \mathbb{S}_F^{T_jT_j} & \mathbb{S}_F^{T_jT_m} \\ \mathbb{S}_F^{T_mT_j} & \mathbb{S}_F^{T_mT_m} \end{pmatrix}, \quad \mathbb{S}_F^{T_pT_q} = \left[\mathcal{S}_F^i(\phi_n^{T_q}, \phi_i^{T_p})\right]_{i\in D_{T_p}, n\in D_{T_q}}, \tag{5.83}$$

where the block $\mathbb{S}_F^{T_pT_q}$ contributes to the block $\mathbb{S}^{T_pT_q}$ of the global stiffness matrix $\mathbb{S}$ for all $p, q \in \{j, m\}$. We can see from (5.81) that a boundary face $F \in \mathcal{F}_h^e$ (i.e. there exist $T_j \in \mathcal{T}$ such that $F = \partial T_j \cap \partial\Omega$) contributes to the diagonal block $\mathbb{S}^{T_jT_j}$ of the global stiffness matrix $\mathbb{S}$, through the local stiffness matrix

$$\mathbb{S}_F^e = \left[\mathcal{S}_F^e(\phi_n^{T_j}, \phi_i^{T_j})\right]_{i,n\in D_{T_j}}. \tag{5.84}$$

Schematically, the approach proposed here to efficiently assembly the global stiffness matrix $\mathbb{S}$, is illustrated in Fig 5.4.

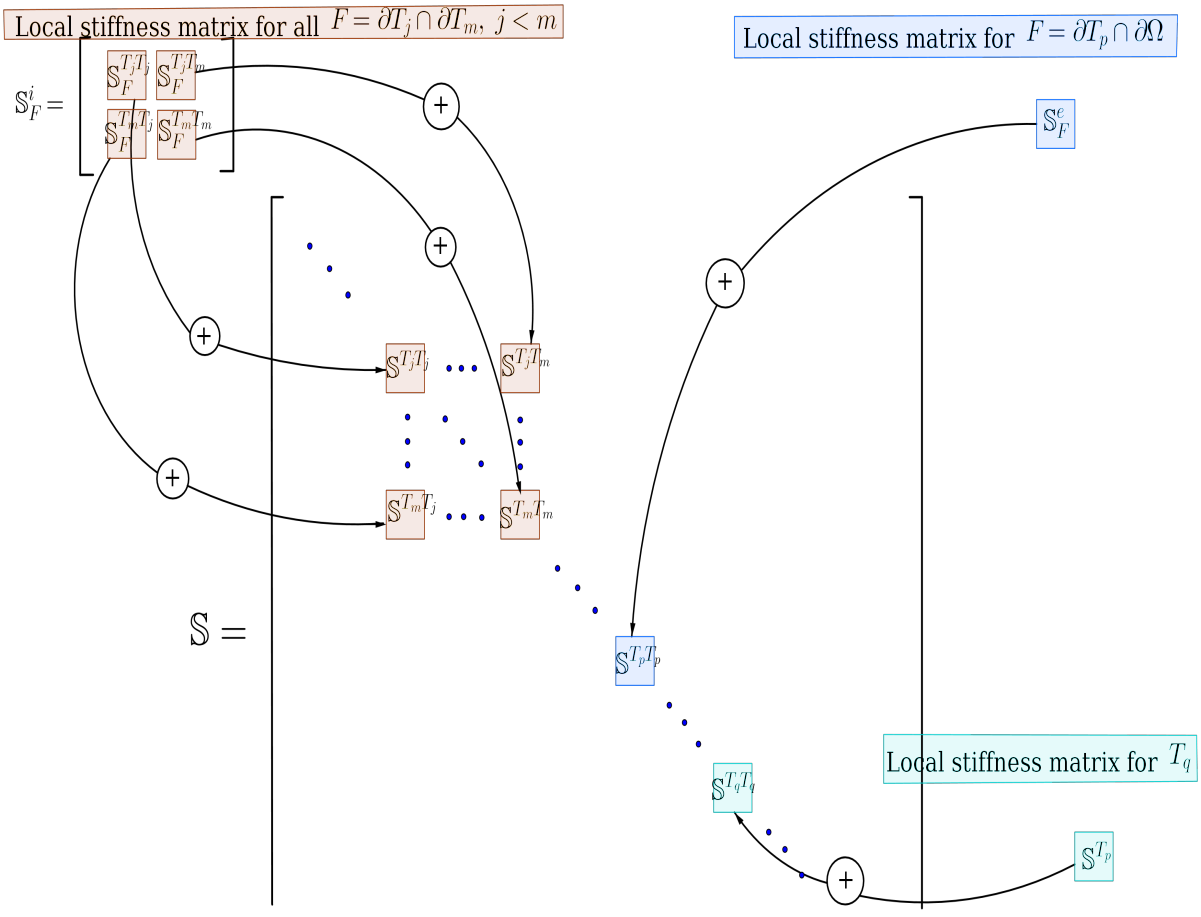


Figure 5.4: Methodology to assemble the global stiffness matrix $\mathbb{S}$.

Finally, let us consider the assembly of the vectors $\mathbb{V}$ and $\mathbb{F}$. These entities can be block-partitioned as follows

$$\mathbb{F} = \begin{pmatrix} \mathbb{F}^{T_1} \\ \mathbb{F}^{T_2} \\ \vdots \\ \mathbb{F}^{T_{N_T}} \end{pmatrix}, \quad \mathbb{V} = \begin{pmatrix} \mathbb{V}^{T_1} \\ \mathbb{V}^{T_2} \\ \vdots \\ \mathbb{V}^{T_{N_T}} \end{pmatrix}, \tag{5.85}$$

where for all $T_j \in \mathcal{T}$, the block vectors $\mathbb{F}^{T_j}, \mathbb{V}^{T_j} \in \mathbb{R}^{N_{dof}^{T_j}}$ are given by

$$\mathbb{F}^{T_j} = \left[\int_{T_j} f \phi_i^{T_j} \right]_{i \in D_{T_j}}, \quad \mathbb{B}^{T_j} = \left[\mathcal{B}(\phi_i^{T_j}) \right]_{i \in D_{T_j}}. \tag{5.86}$$

Note from (5.86) that $\mathbb{F}^{T_j} = \mathbb{M}^{T_j T_j} f^{T_j}$, where f^{T_j} is the coupled components of the restriction of the function f onto the element T_j (i.e. $f\,|_{T_j} \approx \sum_{i=1}^{N_{dof}^{T_j}} f_i^{T_j} \phi_i^{T_j}$). Owing to the localization of basis functions to single mesh elements from (5.70) and the definition of the linear form $\mathcal{B}_F^e$ in (5.68), we have

$$\mathcal{B}_F^e(\phi_i^{T_j}) \neq 0 \Longleftrightarrow F = \partial T_j \cap \partial\Omega, \tag{5.87}$$

for a given basis function $\phi_i^{T_j}$ and an external face F. Therefore a boundary face $F \in \mathcal{F}_h^e$ (i.e. there exist $T_j \in \mathcal{T}$ such that $F = \partial T_j \cap \partial\Omega$) contributes to the block vector $\mathbb{V}^{T_j}$ of the global vector $\mathbb{V}$, locally through

$$\mathbb{V}_F^e = \left[\mathcal{B}_F^e(\phi_i^{T_j}) \right]_{i \in D_{T_j}}. \tag{5.88}$$

In principle, one can evaluate, one by one, the entries of the block matrices $\mathbb{M}^{T_q T_q}$, $\mathbb{S}^{T_q}$, $\mathbb{S}_F^{T_p T_q}$ and $\mathbb{V}_F^e$ by means of suitable quadrature rule. However, such quadrature-based approach negates the advantage that makes the DG method so attractive for implementation on graphics processing units (GPUs). As proposed in [125], we use the 1D and 2D Jacobi polynomials on a reference element to express all non null block matrices in terms of a few global templates (vandermonde and differentiation matrices). As consequence, the implementation methodology proposed here can be

summarised by the following algorithm.

Algorithm 3: Efficient method to assemble $\mathbb{V}, \mathbb{M}$ and $\mathbb{S}$ from DG method.

Initialize $\mathbb{V}, \mathbb{M}$ and $\mathbb{S}$ as sparse matrices
for $q = 1, \cdots, N_T$ do (Loop over elements $T_q \in \mathcal{T}$)
  Compute $\mathbb{M}^{T_qT_q}$ and $\mathbb{S}^{T_q}$ resp. with (5.76) and (5.82)
  $\mathbb{S}^{T_qT_q} \longrightarrow \mathbb{S}^{T_qT_q} + \mathbb{S}^{T_q}$
end for
for $F \in \mathcal{F}_h^i$ i.e. $F = \partial T_j \cap \partial T_m,\ j < m$ do (Loop over internal faces)
  for $p = j, m$ do
    for $q = j, m$ do
      Compute $\mathbb{S}_F^{T_pT_q}$ with (5.83)
      $\mathbb{S}^{T_pT_q} \longrightarrow \mathbb{S}^{T_pT_q} + \mathbb{S}_F^{T_pT_q}$
    end for
  end for
end for
for $F \in \mathcal{F}_h^e$ i.e. $F = \partial T_p \cap \partial\Omega$ do (Loop over external faces)
  Compute $\mathbb{S}_F^e$ and $\mathbb{V}_F^e$ resp. with (5.84) and (5.88)
  $\mathbb{S}^{T_pT_p} \longrightarrow \mathbb{S}^{T_pT_p} + \mathbb{S}_F^e$
  $\mathbb{V}^{T_p} \longrightarrow \mathbb{V}^{T_p} + \mathbb{V}_F^e$
end for

Remark 5.1. In contrast, to assembly the stiffness matrix, Alexandre Ern and Daniele Antonio Di Pietro in [80], follow the same procedure described here by splitting the bilinear function $\mathcal{S}(\cdot,\cdot)$ as follows

$$\mathcal{S}(u_h, w_h) = \underbrace{\sum_{T\in\mathcal{T}} \int_T \{\cdots\}}_{\mathcal{S}_{\mathcal{T}}^0(u_h,w_h)} + \underbrace{\sum_{F\in\mathcal{F}_h^i} \int_F \{\cdots\}}_{\mathcal{S}_{\mathcal{T}}^{i,0}(u_h,w_h)} + \underbrace{\sum_{F\in\mathcal{F}_h^e} \int_F \{\cdots\}}_{\mathcal{S}_{\mathcal{T}}^{e,0}(u_h,w_h)}. \tag{5.89}$$

This leads to the extra CPU time since the support of the each basis function is limited to a single element $T \in \mathcal{T}$. It also leads to an extra contribution from the local stiffness matrix stemming from the internal and external faces.

5.3 Numerical Experiments

The goal in this section is to validate the DG method and its implementation method proposed here for (1.1), through several numerical experiments in two dimensions. As DAREs, we consider: steady advection reaction problem, unsteady diffusion problem, Unsteady Ogata banks problem [177], transport of inert solute trough a

domain with hole and a domain with fracture. The system of ODEs, obtained from the DG discretization of DAREs, is solved with either Impl, ETD, EXPR or ROCK method, as defined in Section 2.3. In order to compare the performance of these time solver combined with DG method, we introduce the L^2-discrete norm, $\| \cdot \|_{L^2(\mathcal{T})}$, given by

$$\| u \|^2_{L^2(\mathcal{T})} = \sum_{T \in \mathcal{T}} \| u \|^2_{L^2(T)} \, . \tag{5.90}$$

We use the linear piecewise polynomials to construct the basis function of the space V_h. To do so, let us consider the triangle T defined by the set of three vertices $\{p_i^T = (x_i^T, y_i^T),\ i = 1, 2, 3\}$. Thus, the corresponding local basis functions $\phi_i^T,\ i = 1, 2, 3$ of T are defined as follow

$$\phi_i^T(x, y) = \begin{cases} a_i^T x + b_i^T y + c_i^T & \forall (x, y) \in T \\ 0 & \forall (x, y) \in \Omega - T \end{cases} , \tag{5.91}$$

where $a_i^T, b_i^T, c_i^T \in \mathbb{R}$ are taken such that $\phi_i^T(x_j, y_j) = \delta_i^j$ for all $j = 1, 2, 3$. The dimension of the finite space is then $\mathcal{N} = 3N_T$, where N_T is the total number of triangles in $\mathcal{T}$. The reason behind the choice of this basis function, is that it enables us to easily handle the projection of any function onto the finite space i.e.

$$\forall u_h \in V_h,\ u_h(x, y) = \sum_{T \in \mathcal{T}_h} \sum_{i=1}^{3} X_i^T \phi_i^T(x, y) \quad \text{with } X_i^T = u(x_i, y_i, t). \tag{5.92}$$

Thus, for the source term $f := f(u_h)$, the vector $\mathbb{F}$ defined in (5.73) is given by $\mathbb{F} = \mathbb{M} f(X)$. As a consequence, all we need to do to solve any DAREs, is to define the bilinear forms $\mathcal{S}_T,\ \mathcal{S}_F^i,\ \mathcal{S}_F^e$ and the linear form $\mathcal{B}_F^e$ associated to the DG formulation of the given DAREs.

5.3.1 Steady advection reaction

The goal here is to validate the upwind DG method proposed for steady advection reaction equation as well as its implementation. To that end, let us consider (5.47) with the convection field $\beta = (2, 2)^t$ and the linear reaction constant $\mu = 1$, by the means of DG method. The source term and the Dirichlet boundary condition are

chosen so that the exact solution is given by

$$u(x, y) = \sin(2\pi x)\sin(\pi y),$$

on the rectangular domain $\Omega = [0, 1] \times [0, 1] \subset \mathbb{R}^2$. Therefore, applying the upwind DG method to (5.47) leads to the system of ODEs

$$\mathbb{S}X = \mathbb{M}f_h, \text{ where } f_h = \left[f(x_i^T, y_i^T)\right]_{i=1,2,3}^{T\in\mathcal{T}}, \tag{5.93}$$

where the matrices $\mathbb{M}$ and $\mathbb{S}$ can be efficiently assembled with Algorithm 3 by considering the bilinear forms $\mathcal{S}_T, \mathcal{S}_F^e$ and $\mathcal{S}_F^e$ define as follow

$$\begin{aligned}
\mathcal{S}_T &:= \mathcal{A}_T^{\mu,\beta}(v_h, w_h) = \textstyle\int_T \mu v_h w_h + (\beta \cdot \nabla_h v_h) w_h, \\
\mathcal{S}_F^i &:= \mathcal{A}_F^{i,\beta}(v_h, w_h) = -\textstyle\int_F \beta \cdot [[v_h]]\{w_h\} - \frac{\eta}{2} \mid \beta \cdot n \mid [[v_h]] \cdot [[w_h]], \\
\mathcal{S}_F^e &:= \mathcal{A}_F^{e,\beta}(v_h, w_h) = \textstyle\int_F (\beta \cdot n)^{\ominus} v_h w_h.
\end{aligned} \tag{5.94}$$

We now demonstrate the decay of the errors between the numerical and exact solutions as we increase the dimension $\mathcal{N}$ of the finite space. This is done by using the uniform refinement of uniform mesh. To construct the uniform mesh, for a given value N, we first subdivide the domain Ω into N^2 squares by subdividing the interval $[0, 1]$ into N uniform intervals in x and y directions. Finally, each squares is split into two triangles. This construction is illustrated in Fig 5.5 for $N = 4$. So the dimension of the finite space is $\mathcal{N} = 6N^2$.

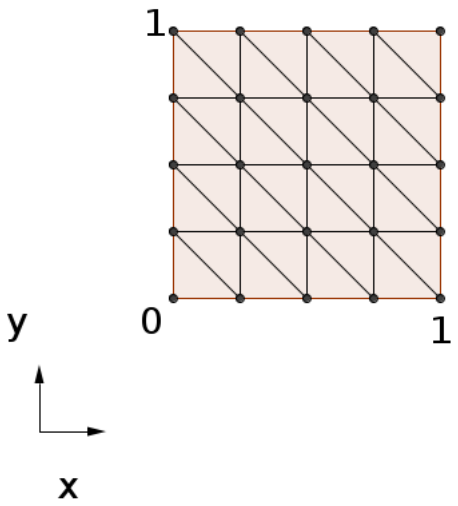


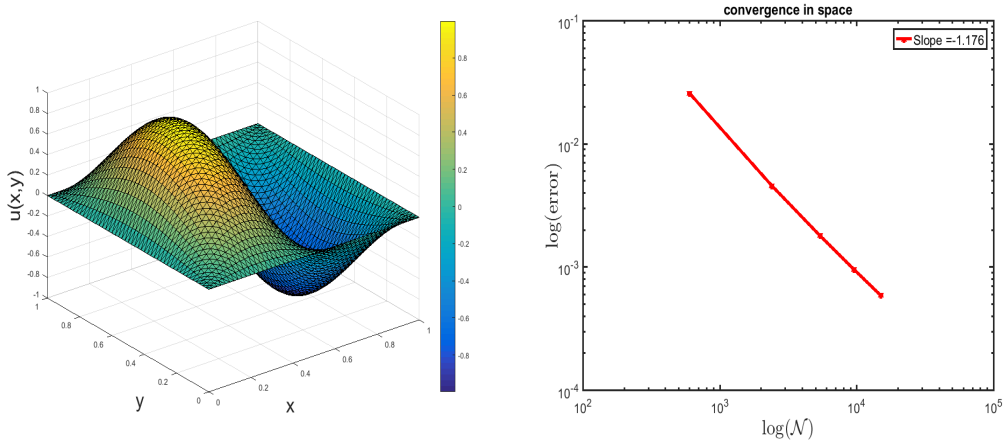
Figure 5.5: Uniform mesh of the domain Ω for $N = 4$ for DG method.

To construct the uniform refinement, for a given integer n, first each square obtained in the process of the construction of the initial uniform mesh, is subdivided into n^2 squares, which are then divide into two triangles. The dimension of the finite space for this refined mesh is then $\mathcal{N} = 6(nN)^2$.

To demonstrate the decay of the errors as we increase the dimension of the finite space, we first construct an initial uniform mesh with $N = 10$. Then for all uniform refinement, $\mathcal{T}_n$ with $n \in \{1, 2, 3, 4, 5\}$, we solve (5.93) and compute L^2-discrete error given by

$$\text{error}_n = \| u - u_h^n \|_{L^2(\mathcal{T}_1)},$$

where u_h^n is the numerical solution associated to the uniform refinement $\mathcal{T}_n$. Finally, we plot in Fig 5.6(a), the numerical solution u_h^5 in three dimension; and we plot in Fig 5.6(b) the $\log(\text{error}_n)$ as a function of $\log \mathcal{N}$. As expected, note from Fig 5.6(b) that the errors between the numerical and exact solutions decrease as we increase the dimension $\mathcal{N}$ of the finite space. It also show that the slope of line is 1.178. Since $\mathcal{N} = 6 \times N^2 = 6 \times (nN)^2 = 6 \times (\frac{n}{h})^2$, thus Fig 5.6(b) shows that the IP-DG scheme, with the piecewise linear polynomials as basis function, is a second order scheme as expected.



(a) (b)

Figure 5.6: **Convergence space of DG method for advection and reaction equation.** We plot in (a), the numerical solution u_h^5 in three dimension. We plot in (b) the $\log(\text{error}_n)$ as a function of $\log \mathcal{N}$. As expected, note from (b) that the errors between the numerical and exact solutions decrease as we increase the dimension $\mathcal{N}$ of the finite space.

5.3.2 Unsteady diffusion

The purpose here is to validate the SIPG method proposed for diffusion equation as well as its implementation. To do so, we investigate the unsteady diffusion respectively with linear and non linear reaction term. In both case, the system of ODEs, obtained from the DG discretisation, is solved with Impl, ETD1, ETD2, EXPR or ROCK2 method, as described in Section 2.3. We compare the performance of SIPG method combines with these time solvers.

First, let us consider the unsteady diffusion with linear reaction equation

$$u_t - \nabla(\epsilon \nabla u) - \lambda u = 0, \tag{5.95}$$

$$u(0, y, t) = u(1, y, t) = u(x, 0, t) = u(x, 1, t) = 0, \tag{5.96}$$

$$u(x, y, 0) = \sin(k\pi x)\sin(j\pi y), \tag{5.97}$$

where $u(x, y, t)$ is a function of two spatial variables $x, y \in [0, 1]$ and the time variable $t \in [0, 1]$; and $\lambda, k, j, \epsilon \in \mathbb{R}^+$. Owing to the Fourier transform (or separation

of variables), we derive the exact solution of (5.95) - (5.97), which is given by

$$u(x,y,t) = \sin(k\pi x)\sin(j\pi y)e^{t(-\epsilon(k^2+j^2)\pi^2+\lambda)}. \tag{5.98}$$

For the simulation in this case, we set $\lambda = 1,\ k = 2,\ j = 2, \epsilon = 10^{-2}$. According to the SIPG method, (5.95) and (5.96) becomes a system of ODEs

$$\mathbb{M}\frac{d}{dt}X + (\mathbb{S} - \lambda\mathbb{M})X = 0, \tag{5.99}$$

where $\mathbb{M}$ is the mass matrix and $\mathbb{S}$ is the stiffness matrix. These entities can be efficiently assembled with Algorithm 3 by considering

$$\begin{aligned}
\mathcal{S}_T &:= \mathcal{D}_T^{\epsilon}(v_h, w_h) = \quad \int_T \epsilon\nabla_h u_h \cdot \nabla_h w_h, \\
\mathcal{S}_F^i &:= \mathcal{D}_F^{i,\epsilon}(v_h, w_h) = \quad -\int_F \{\epsilon\nabla_h u_h\}\cdot[[w_h]] + \{\epsilon\nabla_h w_h\}\cdot[[u_h]] - \tfrac{\eta\epsilon}{h_F}[[u_h]]\cdot[[w_h]], \\
\mathcal{S}_F^e &:= \mathcal{D}_F^{e,\epsilon}(v_h, w_h) = \quad -\int_F \{\epsilon\nabla_h u_h\}\cdot[[w_h]] + \{\epsilon\nabla_h w_h\}\cdot[[u_h]] - \tfrac{\eta\epsilon}{h_F}[[u_h]]\cdot[[w_h]].
\end{aligned} \tag{5.100}$$

We now demonstrate the decay of the errors as we increase the dimension of the finite space or decrease the time step using respectively the space or the time refinement. In space we use the same procedure described in Section 5.3.1 with $N = 10$ and $n = \{2, 3, 4, 5, 6\}$. For each triangulation $\mathcal{T}_n$, we solve (5.99) subject to the initial condition (5.97) (by means of Impl and ETD1) with time fixed time step $dt = 10^{-3}$ and compute the L^2-discrete error

$$\text{error}_n^r = \| u(t=1) - u_h^{n,r}(t=1) \|_{L^2(\mathcal{T}_2)},$$

where $u_h^{n,r}$ is the numerical solution obtained from the combination of SIPG method with the time solver $r \in \{\text{Impl, ETD1}\}$. Throughout the resolution we also store the CPU time, denoted CPU_n^r, associated to each time solver r and triangulation $\mathcal{T}_n$. We respectively plot in Fig 5.7(a) and Fig 5.7(b), the numerical solution at time $t = 1$ for $\mathcal{T}_3$ obtained using the combination of SIPG with Impl and ETD1. We respectively plot in Fig 5.7(c) and Fig 5.7(d), $\log(\text{error}_n^r)$ against $\log(\mathcal{N})$; and $\log(\text{error}_n^r)$ against $\log(\text{CPU}_n^r)$. Note from Fig 5.7(c) that the error decreases as we

increase the dimension of the finite space. It also show that the slope of line is 1.1423 and 1.1762 respectively for ETD1 and Impl integrators. Since $\mathcal{N} = 6 \times N^2 = 6 \times (nN)^2 = 6 \times (\frac{n}{h})^2$, thus Fig 5.7(c) shows that the IP-DG scheme, with the piecewise linear polynomials as basis function, is a second order scheme as expected. Also note from Fig 5.7(d) that, for a given error ETD1 spends less time compared to Impl to simulate the solution at $t = 1$.

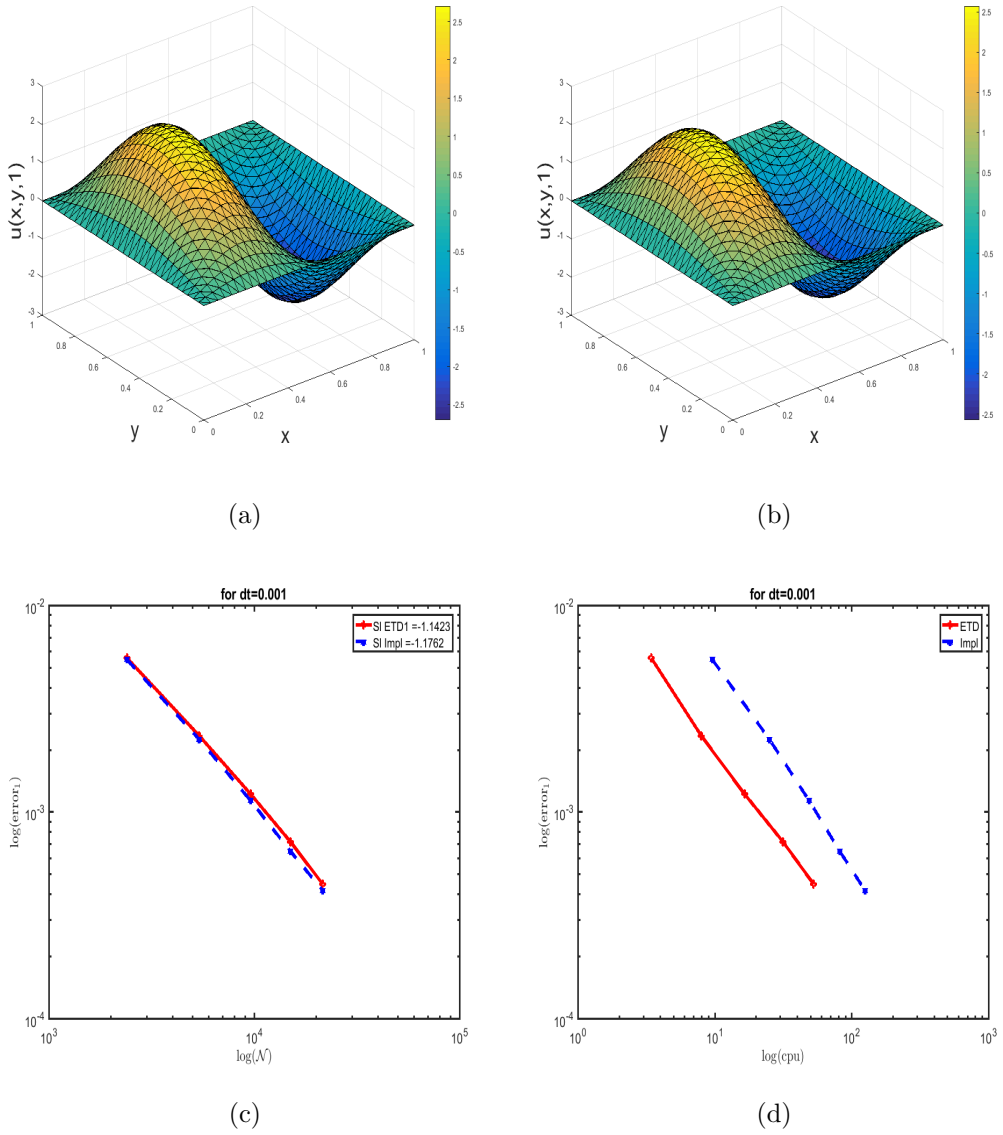


Figure 5.7: **Convergence in space of SIPG method combined with Impl and ETD1 for unsteady diffusion and linear reaction equation.** We respectively plot in (a) and (b), the numerical solution at time $t = 1$ for $\mathcal{T}_3$ obtained using the combination of SIPG with Impl and ETD1. We respectively plot in (c) and (d), $\log(\text{error}_n^r)$ vs $\log(\mathcal{N})$; and $\log(\text{error}_n^r)$ vs $\log(\text{CPU}_n^r)$. Note from (c) that the error decreases as we increase the dimension of the finite space. Also note from (d) that, for a given error, ETD1 spends less time compared to Impl to simulate the solution at $t = 1$.

To illustrate the decay of the error as we decrease the time step of the time solvers, we solve (5.99) subject to the initial condition (5.97) (by means of Impl and ETD1) for the time step $dt_i = 10^{-i}, i \in \{1, 2, 3\}$ on the triangulation $\mathcal{T}_3$. We then compute the L^2-discrete error

$$\text{error}_i^r = \| u(t=1) - u_h^{i,r}(t=1) \|_{L^2(\mathcal{T}_3)},$$

where $u_h^{i,r}$ is the numerical solution associated to the time step dt_i and the solver $r \in \{\text{Impl, ETD1}\}$. We plot in Fig 5.8, $\log(\text{error}_i^r)$ against $\log(dt_i)$. Since (5.99) is linear in X, as expected the error_i^{ETD1} is independent of the time step.

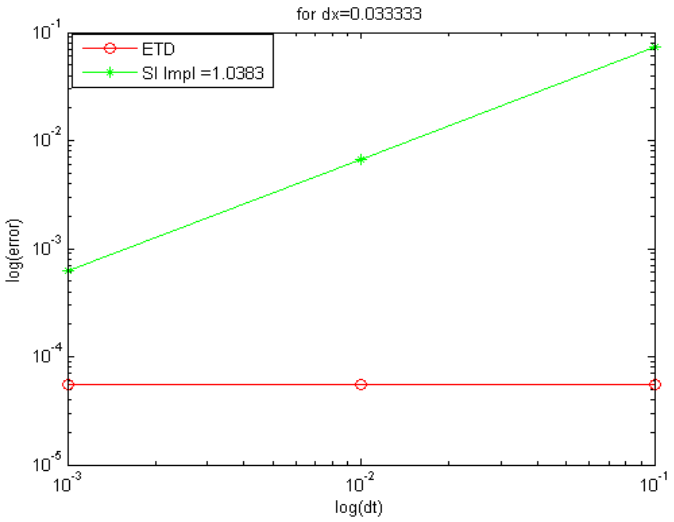


Figure 5.8: **Convergence in time of SIPG method combined with Impl and ETD1 for unsteady diffusion and linear reaction equation.** We plot in this figure, $\log(\text{error}_i^r)$ vs $\log(dt_i)$. As expected, the error_i^{ETD1} is independent of the time step.

Finally, let us examine the unsteady diffusion and non linear reaction equation defined as follows

$$u_t = \nabla(\epsilon \nabla u) + u - u^3, \tag{5.101}$$

$$u(0, y, t) = u(1, y, t) = u(x, 0, t) = u(x, 1, t) = 0, \tag{5.102}$$

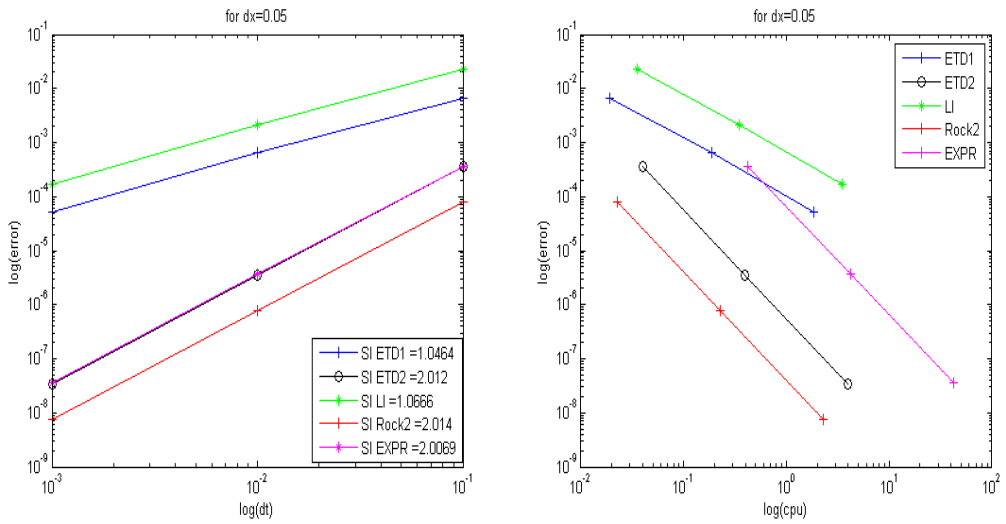
$$u(x, y, 0) = \sin(k\pi x)\sin(j\pi y), \tag{5.103}$$

where $\epsilon = 10^{-4}$, $k = 2$, $j = 1$. By using the SIPG discretization method, (5.101)

and (5.102) becomes a system of ODEs

$$\mathbb{M}\frac{d}{dt}X = (\mathbb{M} - \mathbb{S})X - \mathbb{M}X^3, \tag{5.104}$$

where $\mathbb{M}$ and $\mathbb{S}$ are the same matrices defined for (5.99). To illustrate the decay of the error as we decrease the time step of the time solvers (LI, ETD1, ETD2, EXPR and ROCK2 , as defined in Section 2.3), we use the same procedure described previously. But since the exact solution is unknown in this case, we consider the solution with time step $dt = 10^{-4}$ as the exact solution for each time solvers. We respectively plot in Fig 5.9(a) and (b), $\log(\text{error}_i^r)$ vs $\log(dt_i)$ and $\log(\text{error}_i^r)$ vs log(CPU). Note from Fig 5.9(a) that, for every time solver, the error decreases as we decrease the time step. Fig 5.9(b) show that in the increasing order of efficiency, we have LI<ETD1<EXPR<ETD2<ROCK2.



(a) (b)

Figure 5.9: **Convergence in time of SIPG method combined with LI and ETD1, ETD2, EXPR and ROCK2 for unsteady diffusion and non linear reaction equation.** We respectively plot in (a) and (b), $\log(\text{error}_i^r)$ vs $\log(dt_i)$ and $\log(\text{error}_i^r)$ vs log(CPU). Note from (a) that, for every time solver, the error decreases as we decrease the time step. (b) show that in the increasing order of efficiency, we have LI < ETD1 < EXPR < ETD2 < ROCK2.

Now that the DG method and its implementation method has been validated independently for the diffusion equation and the advection-reaction equation, we now focus on applying the DG proposed analysis to more physical and practical

DAREs in the following sections.

5.3.3 Longitudinal dispersion in porous media

The aim here is to validate the implementation and combined analysis of the upwind DG method and the SIPG method proposed for DAREs. To that end, let us consider the problem of a semi infinite medium having a source plane at $x = 0$, investigated by Ogata and Banks in [177]. In two dimensions, the Ogata and Banks problem is governed by the mass conserved equation

$$\frac{\partial C}{\partial t} - \nabla \cdot (\epsilon \nabla C) + \beta \cdot \nabla C = 0, \quad \beta = (v_x, 0), \tag{5.105}$$

for $\epsilon, v_x \in \mathbb{R}^+$, the time $t \geq 0$ and (x, y) in $\Omega = [0, L] \times [0, 1]$ where L is a positive number, large enough. Initially, saturated flow of fluid of concentration $C = 0$, takes place in the medium. At time $t = 0$, the concentration of the plane source is instantaneously changed to $C = C_0$. Thus, by considering the partition of the boundary $\partial\Omega = \cup_{1=1}^{4} \partial\Omega_i$, schematically illustrated in Fig 5.10.

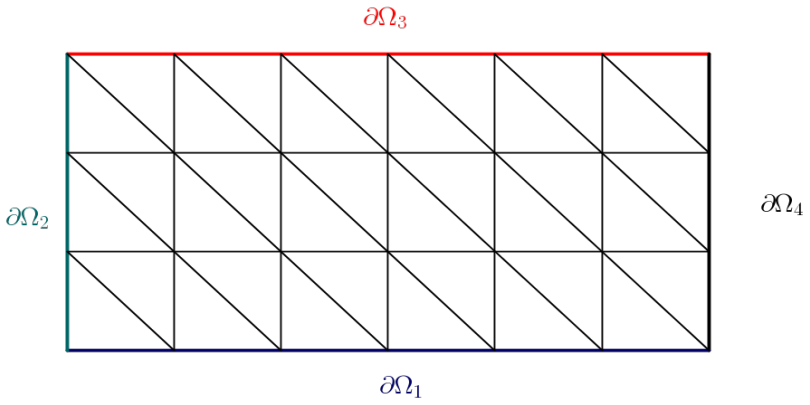


Figure 5.10: Partition of the boundary $\partial\Omega = \cup_{1=1}^{4} \partial\Omega_i$

Then the appropriate boundary condition (BC) and initial condition (IC) are

$$(\text{BC}) : n \cdot \nabla C\Big|_{\partial\Omega_1 \cup \partial\Omega_3} = 0, \quad C\Big|_{\partial\Omega_2} = C_0, \quad C\Big|_{\partial\Omega_4} = 0, \tag{5.106}$$

$$(\text{IC}) : C\Big|_{t=0} = \begin{cases} C_0 & \text{on } \partial\Omega_2 \\ 0 & \text{elsewhere} \end{cases}. \tag{5.107}$$

This problem has been solved analytically by Ogata and Banks in [177], and the solution is given by

$$C(x,y,t) = \frac{C_0}{2}\text{erfc}\left(\frac{x - v_x t}{2\sqrt{\epsilon t}}\right) + \frac{C_0}{2}\exp\left(\frac{v_x x}{\epsilon}\right)\text{erfc}\left(\frac{x + v_x t}{2\sqrt{\epsilon t}}\right). \tag{5.108}$$

By using the DG method proposed for (5.105) subject to (5.106) can be transformed to a system of ODEs

$$\mathbb{M}\frac{d}{dt}X + \mathbb{S}X = \mathbb{B}, \tag{5.109}$$

where the mass matrix $\mathbb{M}$, stiffness matrix $\mathbb{S}$ and the vector $\mathbb{B}$ can be assembled with Algorithm 3, by considering

$$\begin{aligned}
\mathcal{S}_T(u_h, w_h) &= \mathcal{A}_T^{\mu=0,\beta}(v_h, w_h) + \mathcal{D}_T^{\epsilon}(v_h, w_h), \\
\mathcal{S}_F^i(u_h, w_h) &= \mathcal{A}_F^{i,\beta}(v_h, w_h) + \mathcal{D}_F^{i,\epsilon}(v_h, w_h), \\
\mathcal{S}_F^e(u_h, w_h) &= \left[\mathcal{A}_F^{e,\beta}(v_h, w_h) + \mathcal{D}_F^{e,\epsilon}(v_h, w_h)\right] \times q_0, \\
\mathcal{B}_F^e(w_h) &= \int_F \left[(\beta \cdot n)^{\ominus} g w_h - [[g]] \cdot \{\epsilon \nabla w_h\} + \frac{\eta\epsilon}{h_F}[[g]] \cdot [[w_h]]\right] \times q_0,
\end{aligned} \tag{5.110}$$

with the bilinear forms $\{\mathcal{A}_T^{\mu,\beta}, \mathcal{A}_F^{i,\beta}, \mathcal{A}_F^{e,\beta}\}$ and $\{\mathcal{D}_T^{\epsilon}, \mathcal{D}_F^{i,\epsilon}, \mathcal{D}_F^{e,\epsilon}\}$ are respectively given by (5.94) and (5.100). The function g has the same expression as the concentration at the initial time on $\partial\Omega$ and the function q_0 is such that $q_0 = 1$ for all $F \in \partial\Omega_2 \cup \partial\Omega_4$ and $q_0 = 0$ for all $F \in \partial\Omega_1 \cup \partial\Omega_3$. The function q_0 and the linear form $\mathcal{B}_F^e$ appeared here to weakly enforced the mixed boundary conditions of Ogata and Banks problem. In the expression of the bilinear functions $\mathcal{S}_T, \mathcal{S}_F^i, \mathcal{S}_F^e$ and the linear function $\mathcal{B}_F^e$, the term in red and blue respectively handle the advective and diffusion term of (5.105). Since $q_0 = 0$ for all external edges $F \in \partial\Omega_1 \cup \partial\Omega_3$, then the for loop over the external edges in Algorithm 3 is limited the the external edges $F \in \partial\Omega_2 \cup \partial\Omega_4$,

for the seek of efficiency.

To simulate the concentration at the time $t = 1$, we then use ETD1 method to solve (5.109). For the simulations, we consider the velocity component $v_x = 1$ and the concentration $C_0 = 1$, and several values of diffusion coefficient ϵ_i and the constant L_i given by Tab 5.3, thus several Péclet numbers.

i	1	2	3	4
ϵ_i	10^2	10	10^{-1}	10^{-2}
L_i	40	15	2.5	2

Table 5.3: **Settings for Ogata and Banks experiments.** Values of diffusion coefficient ϵ and length L used for the simulation.

We respectively plot in Fig 5.11 (a),(b),(c) and (d) the simulated concentration C^i associated to (ϵ_i, L_i) for $i \in \{1, \cdots, 4\}$ at the time $t = 1$ as a function of the variables x and y. We also plot in Fig 5.11 (e),(f),(g) and (h) respectively the difference between the exact and simulated concentration $C - C^i$ for $i = 1, 2, 3, 4$ at the time $t = 1$ as a function of the variables x and y. As expected, Fig 5.11 shows that $C \approx C_i$ and as we increase the Péclet number (by decreasing the diffusion coefficient) the concentration profile tends to a discontinuous function.

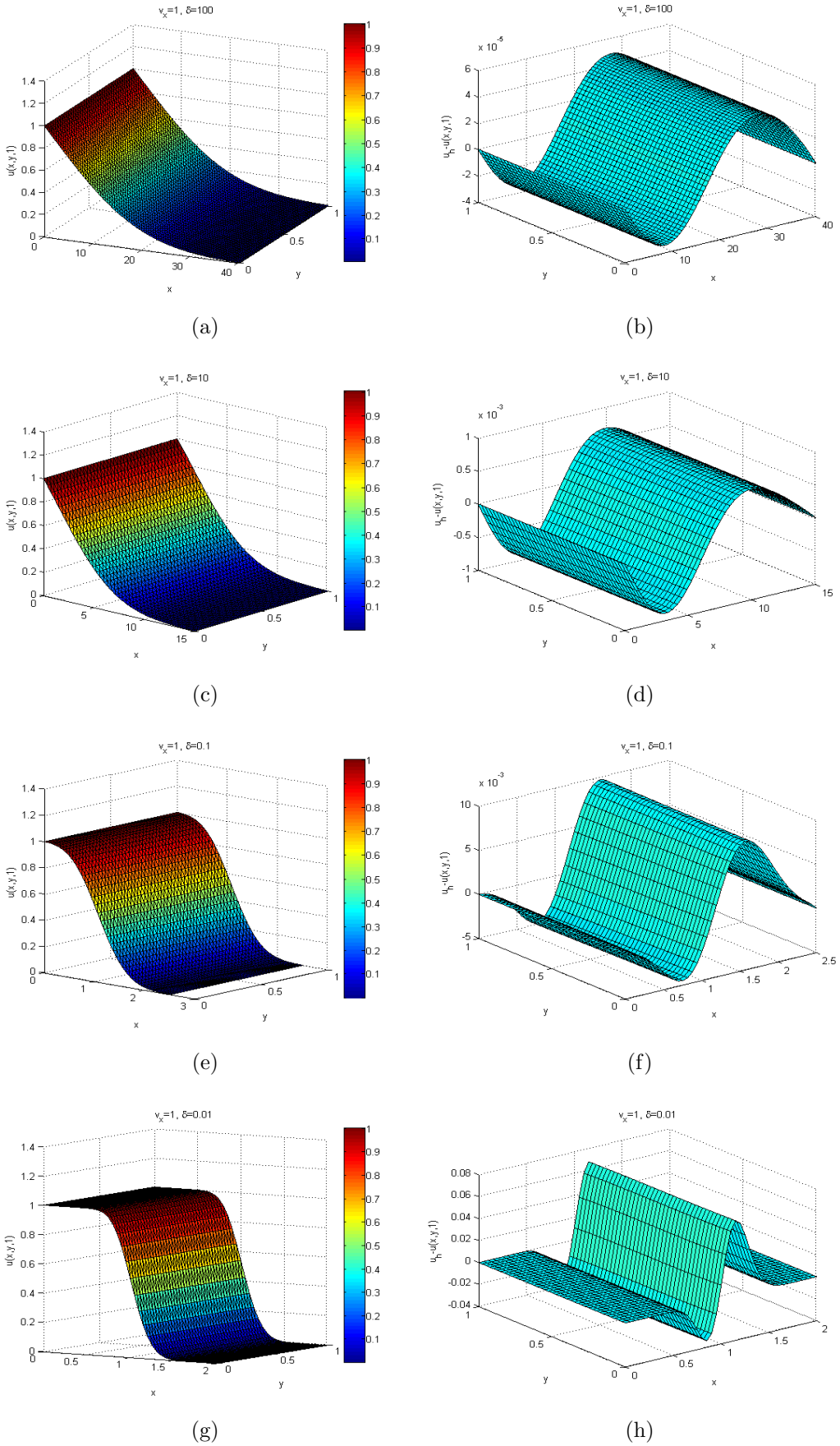


Figure 5.11: **Solution of Ogata and Banks equation for different Pléclet number.** We respectively plot in (a),(b),(c) and (d) the simulated concentration C_i at the time $t = 1$ associated to (ϵ, L) equal $(10^2, 40)$, $(10, 15)$, $(10^{-1}, 2.5)$,$(10^{-2}, 2)$. We also plot in (e),(f),(g) and (h) respectively $C - C_i$ for $i = 1, 2, 3, 4$ at the time $t = 1$.

5.3.4 Transport of solute through a domain with holes

The goal in this section is to use DG method proposed for DAREs and Impl method to investigate transport through more complicated geometry. Let us consider the transport of an inert solute within an incompressible fluid with an absence of volumetric source and sinks, through a two dimension domain, Ω, delimited by two parallel rigid plates. We assumed that the domain Ω contain rigid holes, represented by a finite set of circles, $S_{\mathfrak{C}} = \{\mathfrak{C}_i,\ i = 1, \cdots, n_c\}$, such that $\partial\Omega \cap S_{\mathfrak{C}} = \emptyset$. We recall from Section 5.1.2 and Section 5.1.3, the concentration $C(x, y, t)$ of the solute follows

$$\frac{\partial C}{\partial t} - \nabla \cdot (D_m \nabla C) + \nabla \cdot \mathbf{v}C = 0, \tag{5.111}$$

where D_m is a molecular diffusivity and the velocity $\mathbf{v}$ in each pore, directly given by the solution of Darcy's equation i.e. (5.16) with the equation pressure (5.17).

As a boundary condition, we keep the concentration C and the pressure p respectively at a constant value C_0 and p_0 at the inflow boundary $\partial\Omega_2$ and allow it to undergo pure advection at the outflow boundary $\partial\Omega_4$. The boundary conditions also include the no flux at the rigid boundaries; they are schematically represent in Fig 5.12 with $n_c = 1$. The solute is subject to the initial condition (5.107).

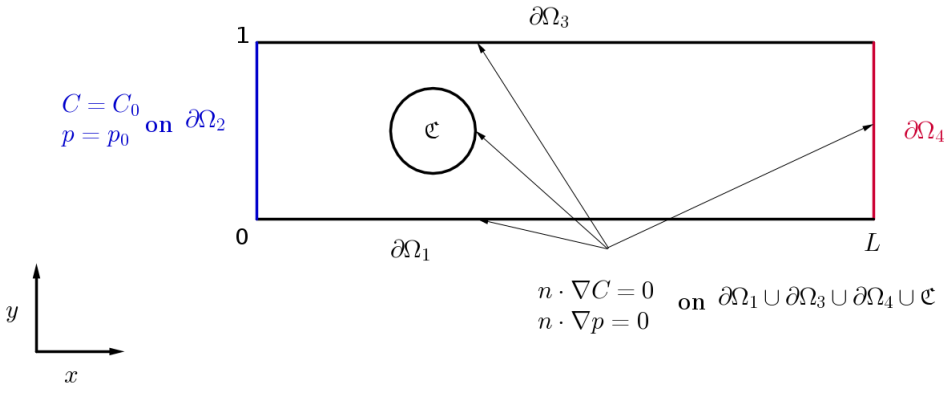


Figure 5.12: Boundary conditions for the concentration and the pressure.

In order to simulate the concentration of the solute in this case, we first set $L = 2$ with $n_c = 6$ and generate unstructured mesh for our domain with the function *distmesh2d*, for more details see [179, 178]. The mesh is designed such that the

elements T get smaller as we get close to the holes. This is illustrated in Fig 5.13.

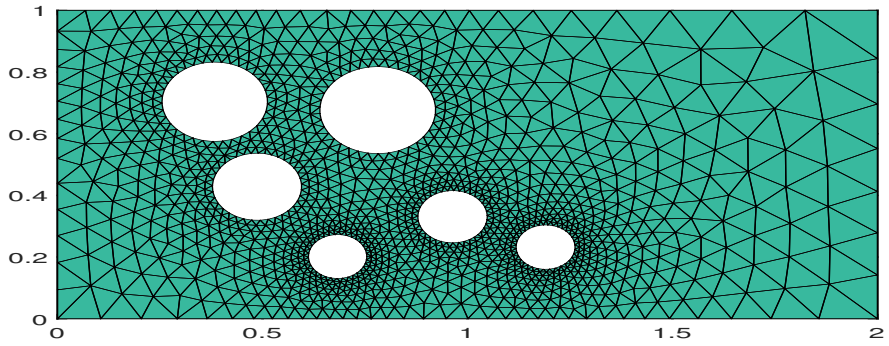


Figure 5.13: Unstructured mesh of a domain with a hole using *distmesh2d*.

Secondly, we consider $p_0 = 1,\ C_0 = 1,\ \phi = 1,\ \mathbf{k}\mu^{-1} = 1$. Then, we use the SIPG method to discretize the steady homogeneous diffusion equation of the pressure, given by (5.17). This leads to the linear system

$$\mathbb{S}_p X_p = \mathbb{B}_p, \tag{5.112}$$

where X_p is the component of the pressure in the finite space V_h; the stiffness matrix $\mathbb{S}_p$ and the vector $\mathbb{B}_p$ are efficiently assembled with Algorithm 3 by considering

$$\begin{aligned}
&\mathcal{S}_T(u_h, w_h) = \mathcal{D}_T^{\epsilon}(v_h, w_h),\\
&\mathcal{S}_F^{i}(u_h, w_h) = \mathcal{D}_F^{i,\epsilon}(v_h, w_h),\ \ \mathcal{S}_F^{e}(u_h, w_h) = \mathcal{D}_F^{e,\epsilon}(v_h, w_h) \times q_1,\\
&\mathcal{B}_F^{e}(w_h) = \int_F \left[-[[g]] \cdot \{\epsilon \nabla w_h\} + \tfrac{\eta\epsilon}{h_F}[[g]] \cdot [[w_h]] \right] \times q_1,
\end{aligned} \tag{5.113}$$

with the set of the bilinear forms $\{\mathcal{D}_T^{\epsilon}, \mathcal{D}_F^{i,\epsilon}, \mathcal{D}_F^{e,\epsilon}\}$ given by (5.100). The function g has the same expression as the pressure on $\partial\Omega_2$ (i.e. $g = p_0$) the function q_1 is such that $q_1 = 1$ for all $F \in \partial\Omega_2$ and $q_1 = 0$ for all $F \in (\partial\Omega \cup S_{\mathfrak{C}}) \setminus \partial\Omega_2$. Thus, the for loop over the external edges in Algorithm 3 is limited to all edges $F \in \partial\Omega_2$.

Once the pressure is simulated, we compute the velocity, $\mathbf{v}^T$, of the fluid on each element T using the expression of the velocity in (5.16). Note that $\mathbf{v}^T$ is constant , since the pressure is approximate with a linear function, for a given element T. We

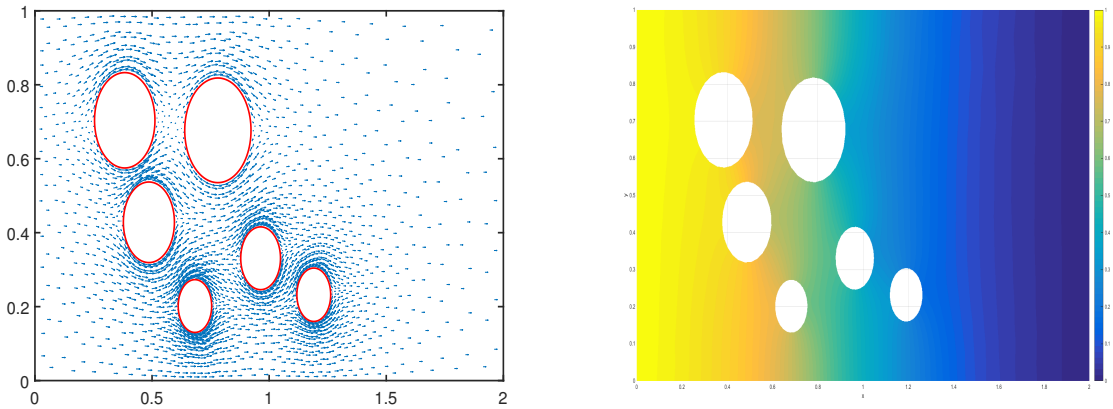
then defined the time step Δt_T for a given element $T \in \mathcal{T}$ as follows

$$\Delta t_T = 2^{-\mathbf{N}^T}, \quad \mathbf{N}^T = \left\lceil \log_2 \left(\frac{\| \mathbf{v}^T \|}{R_T \mathbf{C}_{max}} \right) \right\rceil, \tag{5.114}$$

where $\lceil \cdot \rceil$ is the ceiling function [123], $\mathbf{C}_{max} \in \mathbb{R}$ is the Courant number [67], R_T is the radius of the incircle of the element T. This ensures that the CFL condition is verified on each element T. Thus, to apply the time solver reviewed in Chapter 2, we consider the global time step

$$ht = \min\{\Delta t_T,\ T \in \mathcal{T}\}. \tag{5.115}$$

Finally, we use the DG and Impl method to discretize (5.111) subject to the boundary conditions illustrated in Fig 5.13. This leads to a system of ODEs (5.109) and (5.110) where q_0 here is equal to q_1 (i.e. $q_0 = 1$ on $\partial\Omega_2$ and $q_0 = 0$ on $\partial\Omega \cup S_{\mathfrak{C}}) \setminus \partial\Omega_2$). To simulate the concentration at $t = 1$, we then use here the Impl method (5.109) for $D_m = 10^{-2}\max(\| \mathbf{v}^T \|, T \in \mathcal{T})$ and $\mathbf{C}_{max} = 0.08$. The results are illustrated by plotting in Fig 5.14 (a) the vector field obtained from solving Darcy,s equation. We plot in Fig 5.14 (b), the concentration at the time $t = 1$ as function of x and y.



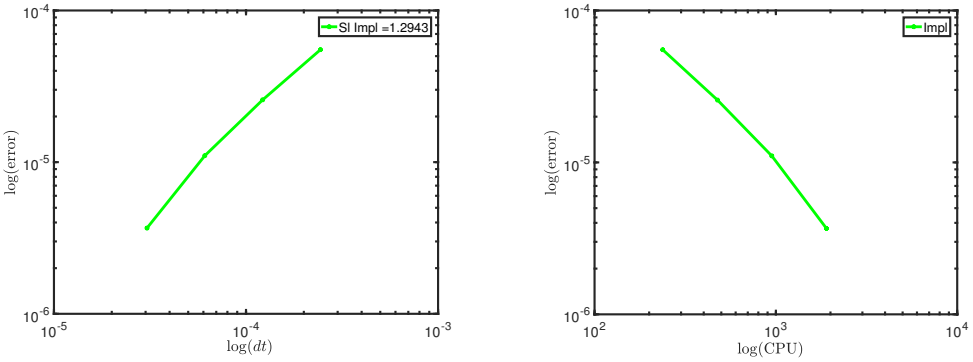
(a) (b)

Figure 5.14: **Transport of solute through a domain with hole.** We plot in (a) the vector field obtained from solving Darcy's equation using DG method. We plot in (b), the concentration at the time $t = 1$, obtained from the DAREs using the DG combined with Impl method, as function of x and y.

Let us now focus on the investigation of the convergence in time of the numerical method, denoted DG_{Impl}, used here (i.e. DG + Impl method). To that end, we simulate the concentration, $C^r_{DG_{Impl}}$, at the time $t = 1$ using the solver DG_{Impl} for all universal time step $ht_r = 2^{-r} \times ht,\ r = 0, \cdots, 4$. Since the exact function is unknown, we assume that it is given by the finest universal time step ht_4 associated to $\mathbf{C}^4_{max}$. We then compute the error, error^r, as follows

$$\text{error}^r_{DG_{Impl}} = \| C^4_{DG_{Impl}} - C^r_{DG_{Impl}} \|_{L^2(\mathcal{T})},\quad r \in \{0, \cdots, 3\},$$

The decay of the error with respect to the time step is illustrated in Fig 5.15 (a) by plotting the errors $\text{error}^r_{DG_{Impl}}$ against the universal time steps ht_r. We then plot in Fig 5.15 (b) the errors $\text{error}^r_{DG_{Impl}}$ against the computation time, $\text{CPU}^r_{DG_{Impl}}$, recorded throughout the simulations.



(a) (b)

Figure 5.15: **Convergence in time of the DG combined with Impl method to simulate the transport of solute through a domain with hole.** We plot in (a) the errors $\text{error}^r_{DG_{Impl}}$ against the universal time steps ht_r. As expected, this shows the decay of the error with respect to the universal time step. We plot in (b) the errors $\text{error}^r_{DG_{Impl}}$ against the computation time, $\text{CPU}^r_{DG_{Impl}}$.

5.3.5 Transport of solute through a domain with fracture

The aim in this section, is to examine the combination of the DG method with the standard time integrators such as Impl, ETD and EXPR for the simulation of the transport of solute through a domain with a fracture. To do so, we first re-conduct the problem described in the previous section but in this case with a domain with

fracture. The fracture is represented by the domain $\Omega_r = [x_r, x_r + l] \times [y_r, y_r - h]$. We assumed that within the fracture, the permeability is 1000 times greater than the permeability of the remaining domain. The boundary condition is schematically illustrated in Fig 5.16.

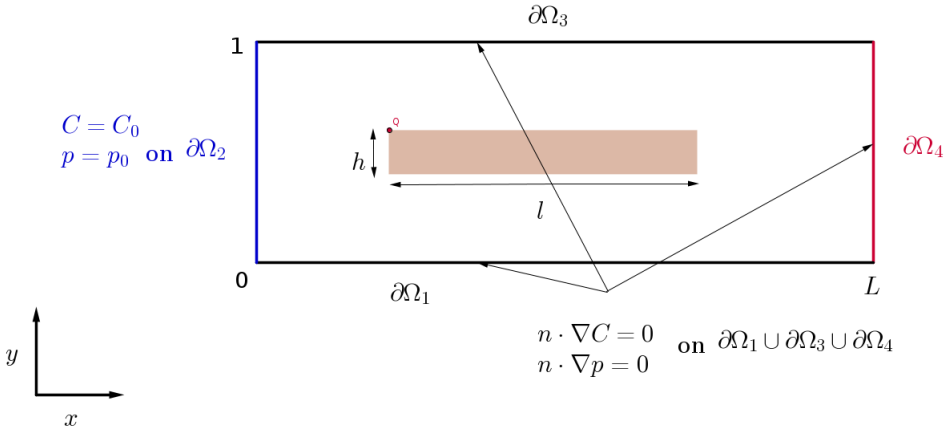


Figure 5.16: Boundary conditions for the concentration and the pressure.

For the simulation of the concentration profile, as in the previous section, we design the unstructured mesh of the domain for $L = 1$ with *distmesh2d*, such that for a given element T within the fracture we have the radius on the incircle of T is less than $h/3$. The finest elements are located in the fracture characterized by the coordinates $x_r = 0.2, y_r = 0.51$, the height $h = 0.02$ and the length $l = 0.6$. This is illustrated in Fig 5.17.

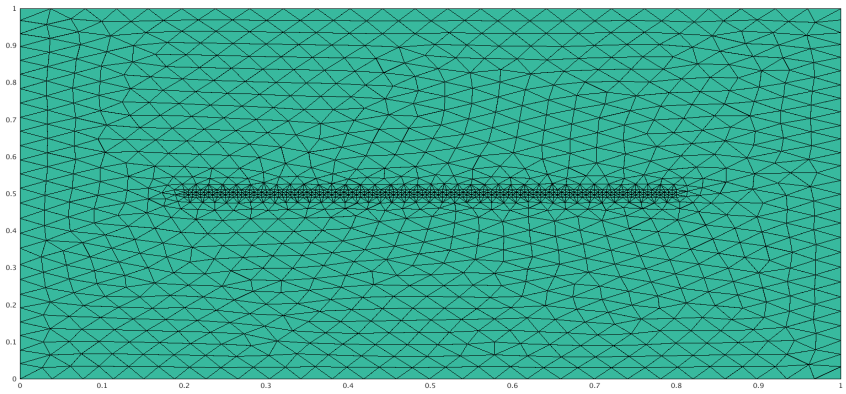


Figure 5.17: Unstructured mesh of domain with fracture using *distmesh2d*.

In this case, the equation of the pressure is a steady heterogeneous diffusion

equation given by (5.17) with

$$p_0 = 1,\ p_1 = 0,\ k\mu^{-1} = \begin{cases} 1000 & \text{on} \quad \Omega_r \\ 1 & \text{on} \quad \Omega \setminus \Omega_r \end{cases} \tag{5.116}$$

We then simulate the fluid velocity on each element T after using the SWIPG method to solve the equation of the pressure. For the sake of clarity, we plot in Fig 5.18 the streamline of the simulated fluid velocity. As expected, Fig 5.18 shows that the velocity of the fluid is higher within the fracture. Once the vector field of the velocity is obtained, we compute the time step Δt_i obtained using (5.114) and (5.115) for all $\mathbf{C}^i_{max} = 0.08 \times 2^{-i},\ i = 0, \cdots, 4$.

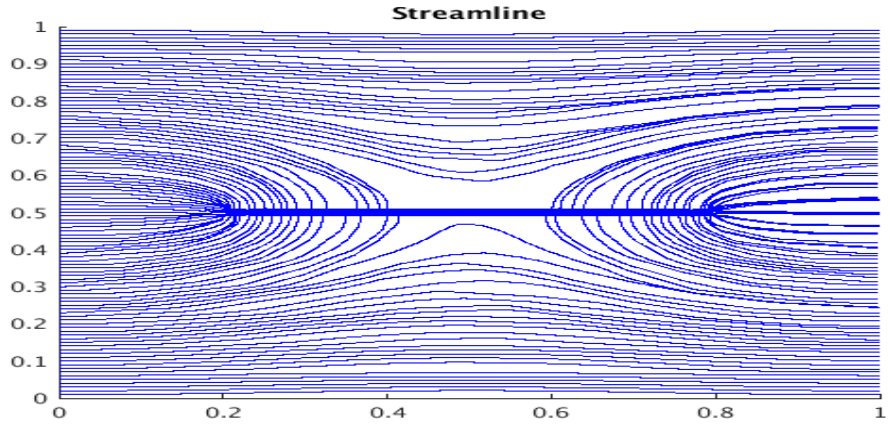
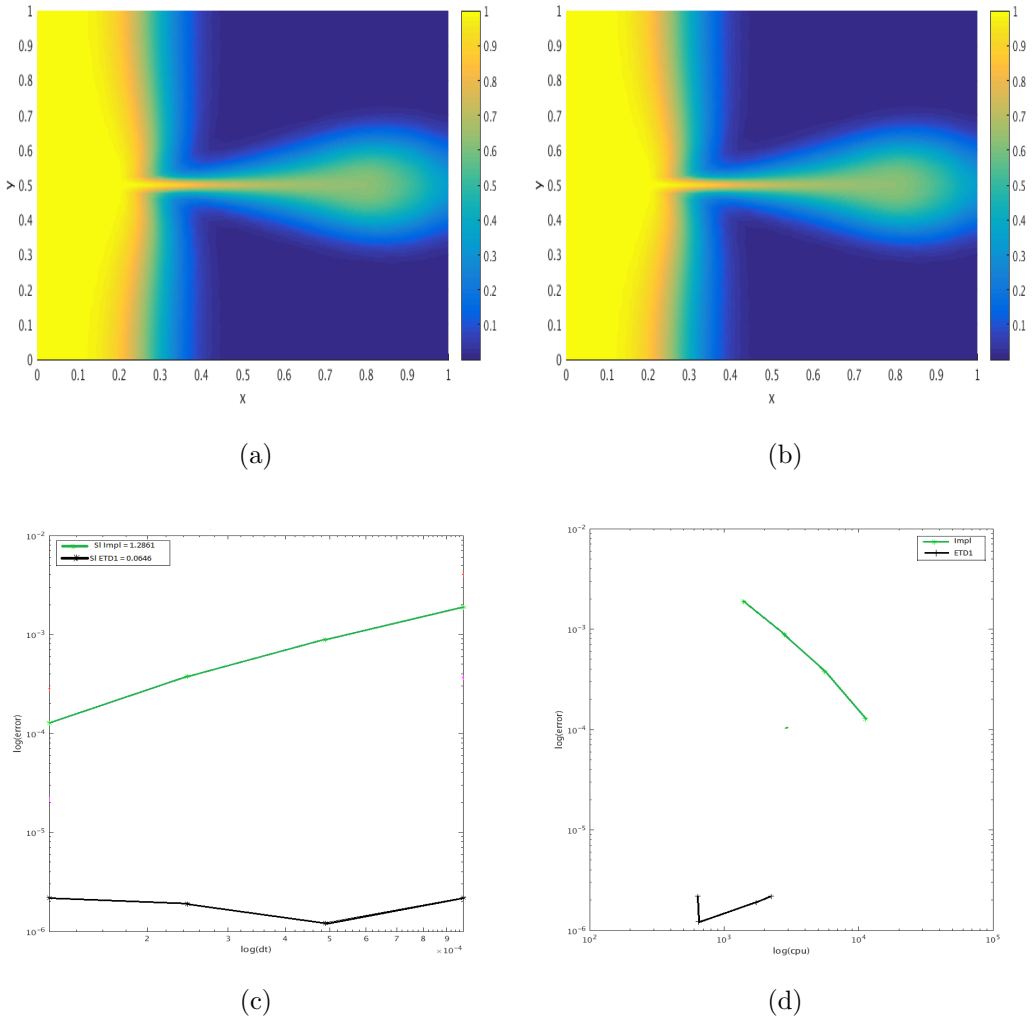


Figure 5.18: Streamline of the simulated fluid velocity of solute through domain with fracture. Note from this figure that the velocity of the fluid is higher within the fracture.

To simulate the concentration at the time $t = 1$ here, we use the DG method to discretize (5.111) subject to the boundary condition illustrated in Fig 5.16. This leads to a system of ODEs (5.109) and (5.110) where $q_0 = 1$ on $\partial\Omega_2$ and $q_0 = 0$ on $\partial\Omega \setminus \partial\Omega_2$, which can be solved with either the Impl or ETD1 method. Thus, to examine the convergence in time, we simulate the concentration, $C^{i,r}$, at the time $t = 1$ for all time step $\Delta t_i, i \in \{0, \cdots, 4\}$ using the time integrators $r =$ Impl, $ETD1$. We then assume that the exact solution is given by $C^{4,r}$ and compute the error as follows

$$\text{error}^{i,r} = \| C^{4,r}(t = 1) - C^{i,r}(t = 1) \|_{L^2(\mathcal{T})}, \tag{5.117}$$

for all $i = 0, \cdots, 3$ and $r =$ Impl, $ETD1$. We respectively plot in Fig 5.19 (a) and (b) the concentration $C^{4,Impl}$ and $C^{4,ETD1}$ as a function of the variables x, y. We also plot the errors $\text{error}^{i,Impl}$ and $\text{error}^{i,ETD1}$ against the time step Δt_i in Fig 5.19 (c). As expected, Fig 5.19 (c) shows that $\text{error}^{i,Impl}$ decays with the time step Δt_i. We plot in Fig 5.19 (d) the errors $\text{error}^{i,r}$ against the CPU time $CPU^{i,r}$ for $r =$ Impl, $ETD1$. Note from Fig 5.19 (d) that ETD1 is more efficient compared to Impl.



(a) (b)

(c) (d)

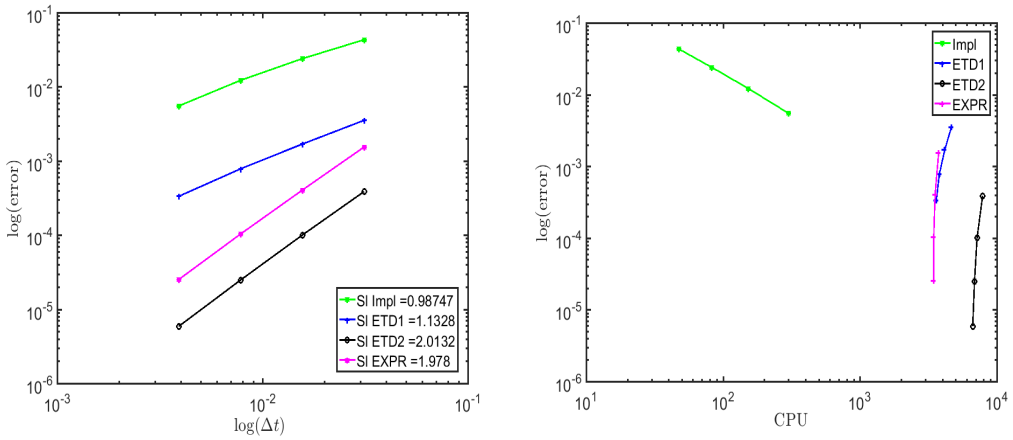
Figure 5.19: **Transport of solute through a domain with fracture without the presence of source and reaction.** We respectively plot in (a) and (b) the concentration $C^{4,Impl}$ and $C^{4,ETD1}$ as a function of the variables x, y. We plot in (c), the errors $\text{error}^{i,Impl}$ and $\text{error}^{i,ETD1}$ against the time step Δt_i. As expected, (c) shows that $\text{error}^{i,Impl}$ decays with the time step Δt_i. We plot in (d) the errors $\text{error}^{i,r}$ against the CPU time $CPU^{i,r}$ for $r =$ Impl, $ETD1$.

Finally, let us consider the case where the source and reaction term is given by

$R(C) = C - C^3$. Then, the governing equation of the solute becomes

$$\frac{\partial C}{\partial t} - \nabla \cdot (D_m \nabla C) + \nabla \cdot \mathbf{v}C = C - C^3. \tag{5.118}$$

Applying the DG method to (5.118) subject to the boundary conditions illustrated in Fig 5.16 leads to (5.104) where the mass and stiffness matrices are the same as in the case of no source term. Then, to simulate the concentration of the solute at the time $t = 1$, we use the time integrators such as Impl, ETD1, ETD2 and EXPR reviewed in Chapter 2 on (5.104), for all Δt_i associated to $\mathbf{C}^i_{max} = 0.5 \times 2^{-i},\ i = 0, \cdots, 4$. We compute the error, $\text{error}^{i,r}$, using (5.117) for all $, i = 0, \cdots, 4$ and $r =$ Impl, ETD1, ETD2 and EXPR. We plot in Fig 5.20(a) the error $\text{error}^{i,r}$ against the time step Δt_i for all $r =$ Impl, ETD1, ETD2 and EXPR. It shows the decays of the error as we decrease the time step. As expected, Fig 5.20(a) shows that Impl and ETD1 are one order time integrators while ETD2 and EXPR are second order. We plot in Fig 5.20(b) the error $\text{error}^{i,r}$ against the CPU time $CPU^{i,r}$ for all $r =$ Impl, ETD1, ETD2 and EXPR. Here, the $CPU^{i,r}$ for all $r =$ ETD1, ETD2 and EXPR are almost constant due to the computation of time of $e^L X$ with *phipm*.



(a) (b)

Figure 5.20: **Transport of solute through a domain with fracture and the presence of source and reaction.** We plot in (a) the error $\text{error}^{i,r}$ against the time step Δt_i for all $r =$ Impl, ETD1, ETD2 and EXPR. It shows the decays of the error as we decrease the time step. As expected, (a) shows that Impl and ETD1 are one order time integrators while ETD2 and EXPR are second order. We plot in (b) the error $\text{error}^{i,r}$ against the CPU time $CPU^{i,r}$ for all $r =$ Impl, ETD1, ETD2 and EXPR.

5.4 Summary

In this chapter, we investigate a numerical method, based on the IP-DG method and the standard time integrators (such as Implicit Euler, ETD, EXPR and ROCK2 methods, see Chapter 2), to simulate the flow and transport in porous media in two and three dimensions. To that end, we revisit the DG space discretization for the DAREs in two or three dimensions, by following the analysis reviwed by Alexandre Ern and Daniele Antonio Di Pietro [80]. In two dimensions, we have applied the combination of the DG discretization with the time integrators such that Impl, ETD, EXPR method, reviewed in Chapter 2 to a variety of linear and non-linear DAREs. This has shown that these numerical methods are suitable for the simulation of the flow and transport in porous media. But even if these numerical methods are reliable, they still expensive for large scale problem i.e. when the dimension of the finite space V_h is large enough (See for example Fig 5.15 (b), Fig 5.20(b)). In the next chapter, this high cost of the proposed numerical method will be investigated by the means of the DG method combines with local time stepping integrators.

Chapter 6

Local time stepping DG methods for DAREs

Contents

The purpose of this chapter is to reduce the computational cost of the numerical methods used in Chapter 3 and Chapter 5, to simulate respectively the cyclic voltammetry models and the transport of solute through the porous media. To that end, we introduce two novel local time stepping discontinuous Galerkin (LTS-DG) schemes for solving efficiently the DAREs on complex geometries. These LTS-DG methods combine domain decomposition techniques, the DG spatial discretization and the standard time integrators such as Impl, ETD, EXPR presented in Chapter 2.

The format of this chapter is as follows. In Section 6.1, we give a brief review of the local time stepping (LTS) method for different types of equations. Next, in Section 6.2, we describe our proposed LTS-DG schemes. Finally in Section 6.3, various numerical results are presented to validate and compare the efficiency of the proposed LTS-DG schemes against GTS-DG schemes used in Chapter 3 and Chapter 5.

6.1 Introduction to LTS

As seen in Chapter 3, the voltammetric response depends only on the local spatial and temporal behaviour of the concentration of the electro-reactive species at the electrode. Moreover, the change of concentration of these species originates from the electrode and then diffuses into the body of domain (see for example Fig 3.5). Also in Chapter 5, we notice a rapid change of the concentration of the solute through the fracture and between the holes (respectively illustrated in Fig 5.19 and Fig 5.14). In general, there are often special features (e.g. fractures, walls, corners, obstacles, electrodes, point loads or irregular material interfaces), which affect locally the flow and transport of solute and as consequence the global solution.

To accurately capture such local behaviour, spatial local refinement is necessary. However, this requires a reduction of the time step, Δt, for stability purpose (while using the explicit time integrator) and for accuracy purposes (while using both explicit and implicit time integrators). Unfortunately, when applied uniformly on all the simulation domain, Ω, the reduced time step leads to an unacceptable large CPU time, making the use of LTS methods highly desirable. The key feature of LTS methods is to split the solution domain Ω into several sub-domains Ω_i each with a time step Δt_i as large as possible for efficiency. According to [103], the LTS method is efficient if it ensures accuracy of the solution, i.e. the solution has to be more accurate than the one obtained with a global coarse mesh, and in addition leads to reduced CPU time compared to the one obtained when using a small time step on the whole domain.

LTS methods have their roots in the work of Rice [193], who in 1960 developped the so-called multirate Runge-Kutta methods for a two scale system of ODEs. The multirate approach, for ODEs, was then combined with linear multistep integrators in 1984 by Gear et al. [115] to improve the accuracy; and their stability properties were analyzed in 1989 by Skelboe et al. [206]. These multirate methods were based on the static partitioning of the domain Ω generated from the priori knowledge of the physics of the problem. This limitation was overcomed in 1997 by Engstler et al. [94] when they introduced the multirate extrapolation methods for ODEs, based on

the Richardson extrapolation. Due to their unconditional stability, the DG method was also used to handle the time refinement of ODEs. The DG method, denoted DG(q) while using the polynomials of degree $q \in \mathbb{N}$, applied to ODEs was first studied in 1974 by LeSaint et al. [154]. It was proven to be strongly A-stable of order $2q+1$ by the authors. Note that the case $q = 0$ is equivalent to implicit Euler scheme. In 1981, another class of DG(q), of order $2q + 2$, appeared in the work of Delfour, Hager et al. [76]. Further earlier work on DG for ODEs can be found in [136, 137]. Adaptive error control was introduced in [141] and more recently considered in [99, 37]. For more insight on the DG method applied to ODEs, see for example [59].

In the case of PDEs, several schemes using the DG method have been developed to handle space and time refinement problems. The FE method in space followed by the DG method in time was used in [4, 55, 54, 95, 96, 100, 198, 199] to solve parabolic problems and extended in [106] to a linear nonstationary convection-diffusion-reaction problem. This method, denoted CG(p)DG(q), used a piecewise polynomials of degree p and q respectively for the space and time discretization. Feistauer et al. [107] proposed the theory of error estimates for CG(p)DG(q) applied to a nonstationary convection-diffusion problem with a nonlinear convection and linear diffusion. The DG method is used in both space and time by Feistauer et al. [50, 108, 85] to solve the nonstationary parabolic problems with nonlinear convection and diffusion.

Other than the DG time discretization, Lörcher et al. [160] used the LTS method (denoted ADER-DG) based on arbitrary high-order derivatives methods and allows every element of the mesh to have its own time-step, which is dictated by the element size. The ADER-DG schemes, as presented for electromagnetism [211] and elastic wave propagation in [90], were obtained by the extension to the DG framework, of the ADER finite volume (ADER-FV) approach which was developed by Toro et al [212]. The ADER-DG scheme was used by Fambri et al. [105] on space-time adaptive meshes for compressible Navier-Stokes equations and the equations of viscous and resistive magnetohydrodynamics in two and three space-dimensions.

Angulo et al. [8] introduced the LTS schemes (denoted LTS-LF) based on the leapfrog (LF) and Runge-Kutta (RK) time integrators, where the mesh is sorted into different sub-domains with appropriate time step on each. In 2009, Diaz and Grote introduced an energy-conserving LTS-LF scheme [82] for the acoustic wave equation, which they extended in 2015 into a multi-level version [83]. Rietmann et al. [194] developed a new LTS method based on the Newmark scheme for large scale wave propagation, which also can be extended to accommodate multiple sub-domains of mesh refinement.

Unfortunately, the DG method in space and time considered in [50, 108, 85, 160, 8, 82, 83, 194] were still special, since they always had boundaries in time aligned with the time direction i.e. the spatial boundaries are independent of the time. To overcome this limitation, an alternative space-time DG method was introduced by van der Vegt et al. in [77, 78] for inviscid compressible flows and extended in [145] to the compressible Navier-Stokes equation. The key feature of this space-time DG method is that no distinction is made between space and time variables and the DG discretization is directly and simultaneously performed in space and time. This then provides flexibility to deal with time dependent boundaries, deforming elements and naturally results in a conservative discretization, even on deforming, locally refined meshes with hanging nodes. A complete hp-error and stability analysis of the space-time DG discretization for the linear advection-diffusion equation is given in [208].

However, the bottleneck of these LTS methods, based on the DG discretization, is that they lead to a large discrete problem in space-time, specially in the presence of complex geometry or localized small-scale physics. As consequence, they can become very expensive in term of storage and computational time. Thus, we focus here on splitting the solution domain on several small regions, yielding several low dimension system of ODEs from the DG spatial discretization, which can then be solved separately. We look at two approaches: the first based on [31, 164, 215, 88, 148, 214] where the sub-domains are overlapped and the second using non-overlapping sub-domains based on [74, 73, 75].

6.2 LTS-DG schemes

The basic idea used in this thesis for the construction of the LTS-DG schemes is to combine domain decomposition techniques, the DG spatial discretization and the time integrators presented in Chapter 2. It follows three basic steps:

1. First, we use a priori knowledge of the local behaviour of the solution due to fractures, walls, corners, obstacles, point loads, etc, to construct a refined mesh of the spatial domain. For example, this is illustrated in Fig 5.17 by showing the refined mesh for the domain with fracture, as described in Section 5.3.5.

2. Secondly, we choose the local time step on each element of the mesh such that it is proportional to the element size (similar to [160]) and inversely proportional to the norm of the fluid velocity (if it is different from zero). This splits the solution domain, Ω, into sub-domains, Ω_i with the local time steps Δt_i for all $i \in \{0, \cdots, m\}$.

3. Finally, we use either interpolation or extrapolation techniques to estimate the solution at the internal boundary, Γ_i, given by

$$\Gamma_i = \partial\Omega_i \setminus \partial\Omega, \tag{6.1}$$

   for all $i \in \{0, \cdots, m\}$. So, one can solve the PDEs independently on each sub-domains, Ω_i, using the DG method combined with the standard time integrators and local time step Δt_i.

We now look at these steps in more details. The first step can be implemented, for example, using the MATLAB's code *distmesh* [179, 178]. We complete the second step by assuming that, beside being proportional to the element size and inversely proportional to the norm of the fluid velocity, the local time step on the sub-domain Ω_i is given by

$$\Delta t_i = \lambda_i \times \Delta t, \tag{6.2}$$

with $\lambda_i \in \mathbb{N}$, for a given $\Delta t \in \mathbb{R}^+$ and all $i \in \{0, \cdots, m\}$. Thus, for a given initial time T^0 and final time T^1, we have a time synchronization over all sub-domains at

a certain time $t \in [T^0, T^1]$ i.e. there exist a time $q_i \in \mathbb{N}$ such that

$$t = q_i \times \Delta t_i, \tag{6.3}$$

for all $i \in \{0, \cdots, m\}$. This is achieved, in two dimension case for example, by using (5.114) which yields the synchronization time $t = t^n$ and decomposition of the solution domain defined as follows

$$\begin{aligned} t^n =& T^0 + n \times \Delta t_{\max}, \quad \text{with} \quad \Delta t_{\max} = \max\{\Delta t_i, i = 0, \cdots, m\}, \\ \Omega =& \bigcup_{i=0}^{m} \Omega_i \quad \text{such that} \quad \Omega_i = \bigcup_{p=1}^{N_i} T_p, \quad \text{with} \quad N_i \in \mathbb{N}, T_p \in \mathcal{T}_h. \end{aligned} \tag{6.4}$$

The sub-domains obtained when we applied the second step to the fracture problem (described in Section 5.3.5) is illustrated in Fig 6.1. Note from Fig 6.1 that $m = 2$ with the sub-domains Ω_0, Ω_1 and Ω_2 respectively represented by the color black, blue and yellow. One can also see that the sub-domain Ω_1 is the union of two disjoint regions while its interior boundary Γ_1 is shared with the sub-domains Ω_0 and Ω_2.

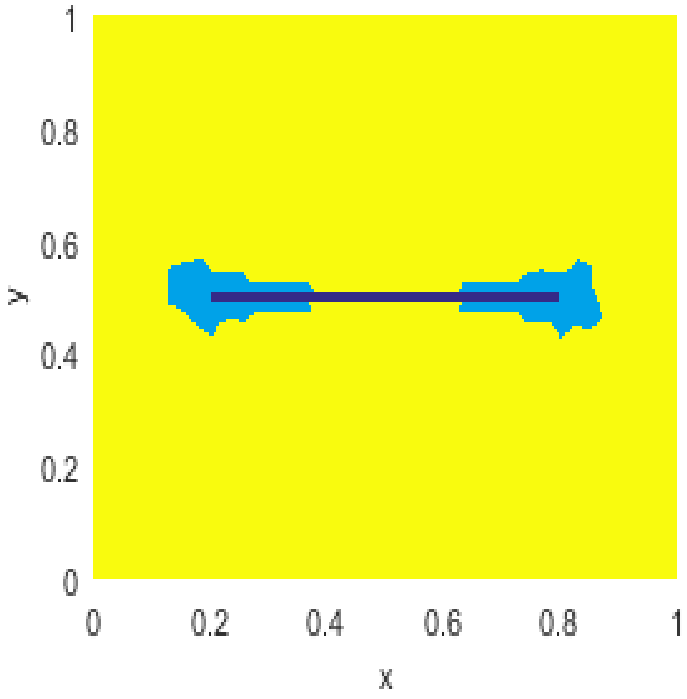


Figure 6.1: Sub-domains: Ω_0(black), Ω_1(blue) and Ω_2(yellow).

Once the sub-domains are defined, the large system of ODEs (5.72), obtained from the DG space discretization of the DAREs, can then be split into $m+1$ smaller systems of ODEs, denoted SO_i for all $i \in \{0, \cdots, m\}$, given by

$$(\mathrm{SO}_i): \quad \mathbb{M}_i \frac{d}{dt} X_i + \mathbb{S}_i X_i = \mathbb{F}_i + \mathbb{B}_i + \mathbb{S}_i^e X_i\Big|_{\Gamma_i} \quad \text{on} \quad \Omega_i \times [0, T]. \tag{6.5}$$

Here, X_i, $\mathbb{S}_i, \mathbb{M}_i$, $\mathbb{F}_i$ and $\mathbb{B}_i$ respectively represent the local solution, stiffness matrix, the mass matrix, source term and the contribution of the global boundary condition on the sub-domain Ω_i. The matrix $\mathbb{S}_i^e$ is used to weakly enforce the internal boundary condition. Therefore, the sum of the last two terms at the right hand side of (6.5) enforces the local boundary condition on $\partial\Omega_i$. Because the IP-DG method is compact and the global mass matrix obtained from the IP-DG spatial discretization is either block-diagonal or diagonal, then the local entities X_i, $\mathbb{S}_i, \mathbb{M}_i$, $\mathbb{S}_i^e$, $\mathbb{F}_i$ and $\mathbb{B}_i$ can be easily extracted from their global values defined in Section 5.2.4. An example of this extraction is shown in Section 6.3.2.

Remark 6.1. Let us introduce the following notation

$$t_i^j = T^0 + j \times \Delta t_i, \qquad s_i = \frac{T^1 - T^0}{\Delta t_i}, \qquad X_i^j = X_i\Big|_{t=t_i^j}, \tag{6.6}$$

for all $i \in \{0, \cdots, m\}$ and all $j \in \{0, \cdots, s_i\}$. If one can estimate $X_i^j\Big|_{\Gamma_i}$, by any means, then the local solution X_i^j at time t_i^j can be obtained from its initial value X_i^0, by applying the time integrator schemes (described in Chapter 2) to the local system SO_i given by (6.5), with the uniform local time step Δt_i.

As a consequence of Remark 6.1, the construction of our LTS-DG schemes is reduced to finding a way to estimate the component $X_i^j\Big|_{\Gamma_i}$ at any time t_i^j for all $i \in \{0, \cdots, m\}$ and all $j \in \{0, \cdots, s_i\}$. We only need to describe the LTS-DG schemes to advance the global solution X, on the solution domain Ω, from its known value at the synchronized time t^n (as it happens at the initial time T^0) to the synchronized time t^{n+1}. The process can then be repeated, in order to estimate the global solution X at the final time T^1 from the global solution X^0 at the initial time T^0. Next, in Section 6.2.1 and Section 6.2.2, we described two different techniques

respectively refer as overlap LTS-DG and non-overlap LTS-DG methods to locally advance the solution from synchronized time t^n to t^{n+1}.

6.2.1 Overlap LTS-DG schemes (OLTS-DG)

The key idea of the overlap LTS-DG methods, OLTS-DG, proposed here is to overlap the different sub-domains obtained from the decomposition of the solution domain Ω, in order to extrapolate the components $X_i^j\Big|_{\Gamma_i}$ at any time t_i^j for all $i \in \{0, \cdots, m\}$ and all $j \in \{0, \cdots, s_i\}$. This approach appeared in [31], where a Crank-Nicolson scheme was used for the time-space discretization of the one spatial dimension heat equation. The authors proved that without local refinement in time ($\Delta t_i = \Delta t$) and space ($h_i = h$) this scheme is stable, provided that

$$\Delta t \leq C \left(\frac{L}{\log L} \right)^2 h^2,$$

where Lh for $L \in \mathbb{N}$ is the size of the overlap, and an error estimate of the form $O(\Delta t^2 + h^2)$. So, in this case, increasing the size of the overlap can reduce the stability constraint on the time step. To avoid the stability constraint, Ewing et al. [102] used a standard centred finite difference scheme in space with backward Euler in time for a linear DAREs. More recently, an approach based on domain decomposition and finite volume discretization, has been proposed by Faille et al. [104] for the one dimensional heat equation. It was used by Gander et al. [114] to investigate the one dimensional convection dominated nonlinear conservation laws.

Here, we extend this approach to the DAREs in one, two or three spatial dimensions, by using the DG method for the space discretization and time integrators such as Impl, ETD, EXPR for the resolution in time of (6.5), in order to avoid the instability.

Overlapping procedure of the domain solution

Once the sub-domains $\Omega_i,\ i \in \{0, \cdots, m\}$ are obtained, we overlap them by pushing the internal boundary Γ_i in the direction of the outward normal vector. This is schematically illustrated in Fig 6.2, for given two sub-domains Ω_0 and Ω_1. The

initial internal boundary is represented by the red line.

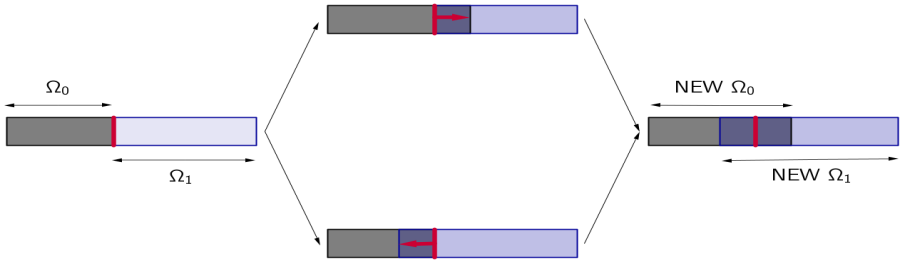


Figure 6.2: The procedure to overlap the regions for $\Omega = \Omega_1 \cup \Omega_2$. On the left: original sub-domains Ω_1 and Ω_2. Middle: we push the internal boundary in the direction of the outward nornal vector. Right: new sub-domains Ω_1 and Ω_2

During the overlapping procedure, if any new sub-domain Ω_i swallows entirely another initial sub-domain Ω_j, then we set $\Delta t_j = \Delta t_i$ so that the sub-domain Ω_j will be included in Ω_i. Later on, in Section 6.3.1, we investigate numerically the effect of the size of the overlap on the accuracy of our OLTS-DG schemes.

Let us denote $\Gamma_{i,j}$ the part of the internal boundary Γ_i included in the sub-domain Ω_j with $i \neq j$ (i.e. $\Gamma_{i,j} = \Gamma_i \cap \Omega_j$). This is schematically illustrated in Fig 6.3. It shows the overlapped sub-domains Ω_i, Ω_j and also the internal boundary $\Gamma_{i,j}$ in red dash. In this case, every time we advance the local solution X_j on Ω_j, we can update the $X_i\Big|_{\Gamma_{i,j}}$.

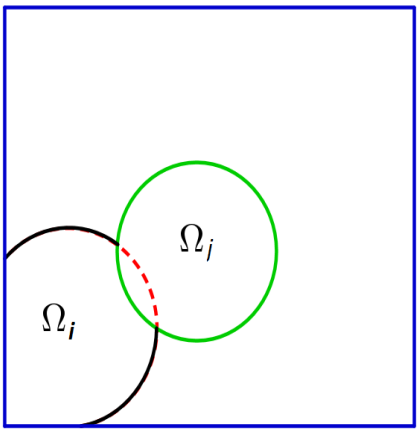


Figure 6.3: Two overlapped sub-domains, Ω_i and Ω_j, showing the internal boundary $\Gamma_{i,j}$ with a red dash line. On the internal boundary $\Gamma_{i,j}$, The solution is updated everytime we advance the solution locally on Ω_j.

Definition 6.1 (Eligible sub-domains)**.** For a given $r \in \{1, \cdots, \frac{\Delta t_{\max}}{\Delta t}\}$, the set of eligible sub-domain S^r is the set of sub-domains on which the known solution has to be advanced locally to the time $t^{n_r} = t^n + r \times \Delta t$. Then we have

$$S^r = \left\{ \Omega_i \middle| \exists j \in \mathbb{N}, t_i^j = t^{n_r} \right\}. \tag{6.7}$$

Note that there is a freedom in the order of which the sub-domains are updated. For example it could either be in the increasing or decreasing order of local time step. If the time integrator $INT \in \{\text{Impl, ETD, EXPR }\}$ is used to advance locally the solution, we denote $DG_{\text{OLTSD-INT}}$ and $DG_{\text{OLTSI-INT}}$ the OLTS-DG scheme that updates the solution on the eligible sub-domains S^r respectively in the decreasing and increasing order of the local time step. In Section 6.3.1, we compare the accuracy of the $DG_{\text{OLTSD-Impl}}$ and $DG_{\text{OLTSI-Impl}}$ and examine how the direction of the bulk velocity of the DAREs or the size of the overlap affect their accuracy. Unless stated, the OLTS-DG method considered for the numerical experiments is the $DG_{\text{OLTSD-INT}}$. Next, we describe step by step the algorithm of the OLTS-DG scheme, $DG_{\text{OLTSD-INT}}$, in order to advance the solution from a synchronized time t^n to t^{n+1}.

Description of the OLTS-DG algorithm

In this section, we describe step by step the algorithm of the OLTS-DG scheme, $DG_{\text{OLTSD-INT}}$, in order to advance the solution from a synchronized time t^n to t^{n+1}. To that end, we consider the overlapped sub-domains illustrated in Fig 6.4, where the solution domain is split into three different sub-domains (i.e. $m = 2$) with the coefficient λ_i in (6.2) given by $\lambda_i = 2^i$ for all $i = 0, \cdots, m$. The overlapped sub-domains Ω_0, Ω_1 and Ω_2 are respectively represented by the color red, blue and green. Note from Fig 6.4 that the internal boundaries are given by $\Gamma_0 = \Gamma_{0,1}$, $\Gamma_1 = \Gamma_{1,0} \sqcup \Gamma_{1,2}$, $\Gamma_2 = \Gamma_{2,1}$. For a given time t^{n_r}, let us denote $\mathbb{X}_i^{n_r}$ and $\mathbb{X}_{i,j}^{n_r}$ the restriction of the global solution X respectively to the internal boundaries Γ_i and $\Gamma_{i,j}$ i.e.

$$\mathbb{X}_i^{n_r} = X\Big|_{\Gamma_i, t=t^{n_r}}, \quad \mathbb{X}_{i,j}^{n_r} = X\Big|_{\Gamma_{i,j}, t=t^{n_r}}. \tag{6.8}$$

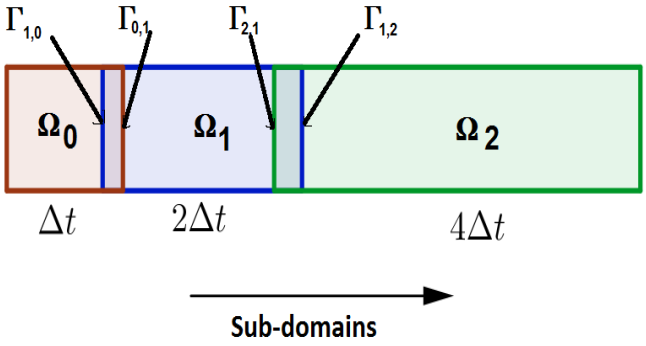


Figure 6.4: Overlaped sub-domains of the solution domain.

Now let discuss step by step, how to update the solution on these interior boundaries in order to advance the solution from the synchronized time t_n to t_{n+1}.

- **Step one:** for $r = 1$, the set of eligible sub-domains is given by $S^1 = \{\Omega_0\}$ and requires $\mathbb{X}_0^{n_1}$ to advance locally on Ω_0 from t^n to $t^{n_1} = t^n + \Delta t$. To that end, we then use the extrapolation $\mathbb{X}_0^{n_1} = \mathbb{X}_0^n$. The completion of this step defines $\mathbb{X}_{1,0}^{n_1}$, as illustrated in Fig 6.5.

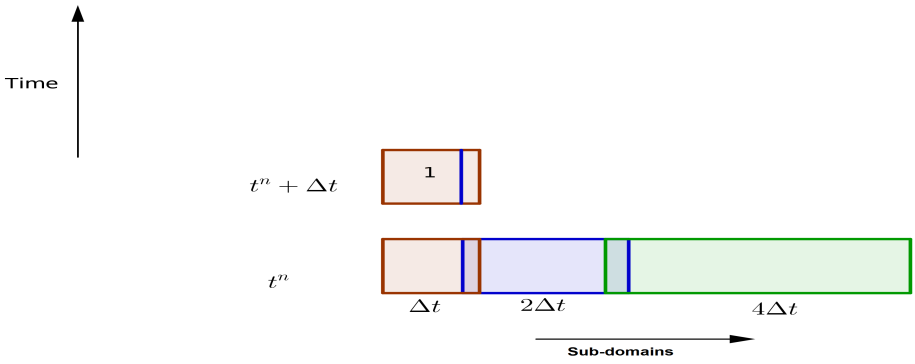


Figure 6.5: The eligible solution advanced to t^{n_1}.

- **Step two:** for $r = 2$, the set of eligible sub-domains is given by $S^2 = \{\Omega_0, \Omega_1\}$, and we require $\mathbb{X}_0^{n_2}$ and $\mathbb{X}_1^{n_2}$ to locally advance the solution to $t^{n_2} = t^n + 2 \times \Delta t$. So we first use the extrapolations $\mathbb{X}_{1,0}^{n_2} = \mathbb{X}_{1,0}^{n_1}$ and $\mathbb{X}_{1,2}^{n_2} = \mathbb{X}_{1,2}^{n}$ to locally advance the solution on Ω_1 from t^n to t^{n_2}. Note from Fig 6.6 that the completion of this simulation defines $\mathbb{X}_0^{n_2}$ and $\mathbb{X}_2^{n_2}$. We finally use $\mathbb{X}_0^{n_2}$ to advance locally the solution on Ω_0 from t^{n_1} to $t = t^{n_2}$.

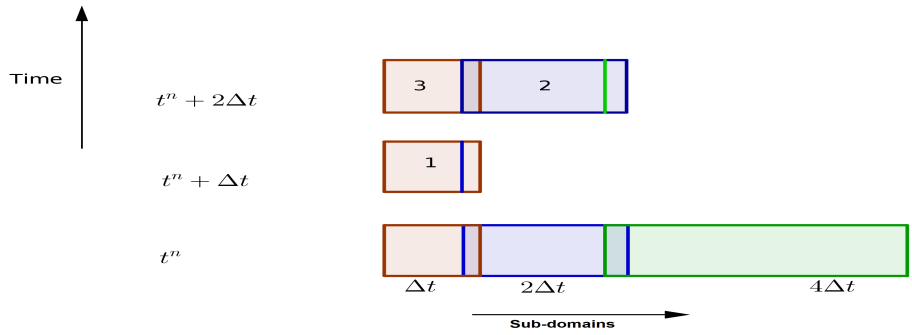


Figure 6.6: The eligible solution advanced to t^{n_2}.

- **Step three:** for $r = 3$, the set of eligible sub-domains is given by $S^3 = \{\Omega_0\}$ and require $\mathbb{X}_0^{n_3}$ to advance locally on Ω_0 to $t^{n_3} = t^n + 3 \times \Delta t$. We use the extrapolation $\mathbb{X}_0^{n_3} = \mathbb{X}_0^{n_2}$. This is illustrated in Fig 6.7 and it shows that the completion of this step defines $\mathbb{X}_{1,0}^{n_3}$.

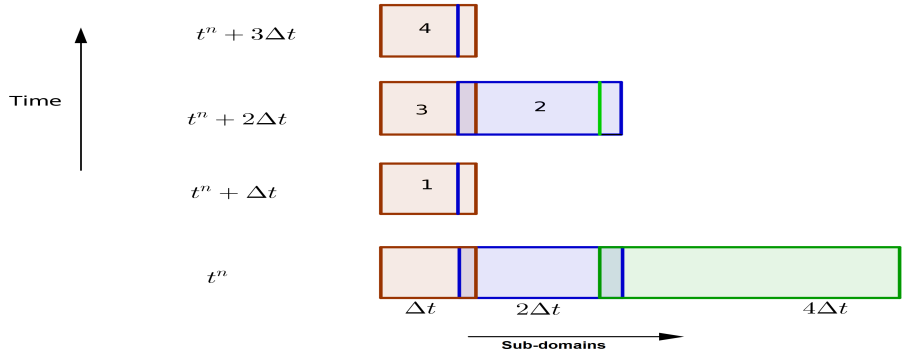


Figure 6.7: The eligible solution advanced to t^{n_3}.

- **Step four:** for $r = 4$, the set of eligible sub-domains is given by $S^4 = \{\Omega_0, \Omega_1, \Omega_2\}$, and we require $\mathbb{X}_0^{n_4}, \mathbb{X}_1^{n_4}$ and $\mathbb{X}_2^{n_4}$ to locally advance the solution to $t^{n_4} = t^n + 4 \times \Delta t$. We first locally advance the solution on Ω_2 from t^n to $t = t^{n_4}$, using the extrapolation $\mathbb{X}_2^{n_4} = \mathbb{X}_2^{n_2}$. This is illustrated in Fig 6.8 and it shows that this simulation defines $\mathbb{X}_{1,2}^{n_4}$. We then use $\mathbb{X}_{1,2}^{n_4}$ obtained and

the extrapolation $\mathbb{X}_{1,0}^{n_4} = \mathbb{X}_{1,0}^{n_3}$, to locally advance the solution on Ω_1 from t^{n_2} to t^{n_4}. This is also illustrated in Fig 6.8. It shows that this simulation defines $\mathbb{X}_0^{n_4}$, which we then use to locally advance on Ω_0 from t^{n_3} to t^{n_4}.

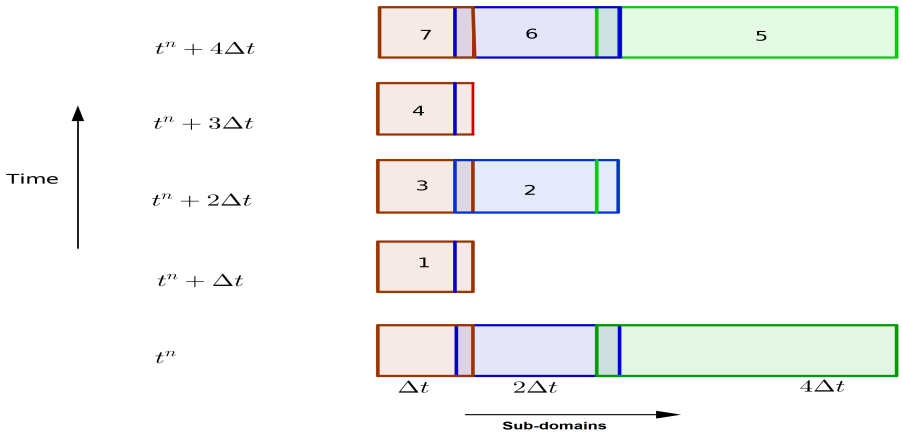


Figure 6.8: The eligible solution advanced to t^{n_4}.

At this point, the time is synchronized across the whole domain Ω. By repeating this process (i.e. from step one to step four) $m^s = \frac{T^1 - T^0}{\Delta t_{\max}}$ times, we can estimate the solution at the final time T^1, from the solution at the initial time T^0. One can implement Algorithm 4, where S is the set of all overlapped sub-domains Ω_i, $\mathbf{E}$ and $\mathbf{A}_i$ respectively represent the extrapolation procedure and the iterative function of the standard time integrators used to solve the local system SO_i.

Algorithm 4: Pseudo algorithm of the overlap LTS method.

for $n = 0, \cdots, m^s - 1$ do (Advance the solution from T^0 to T^1);
  $t^n = T^0 + j \times \Delta t_{\max}$; (Computation of synchronized time t^n)
  $t_i^{known} = t^n \ \ \forall i \in \mathbf{S}$; (initialize the time of the known X_i)
  for $r = 1, \cdots, \frac{\Delta t_{\max}}{\Delta t}$; (Advance the solution from t^n to t^{n+1})
    $t^{n_r} = t^n + r \times \Delta t$; (Computation of t^{n_r})
    for $\Omega_i \in S^r$; (Loop over the eligible sub-domains)
      $\mathbb{X}_i^{n_r} = \mathbf{E}\left(X_l\Big|_{t_{known}^l}, \Omega_l \in \mathbf{S} \setminus \{\Omega_i\}\right)$; (Extrapolate $\mathbb{X}_i^{n_r}$)
      $X_i\Big|_{t^{n_r}} = \mathbf{A}_i\left(X_i\Big|_{t_i^{known}}, \mathbb{X}_i^{n_r}\right)$; (Advance solution on Ω_i to t^{n_r})
      $t_i^{known} = t^{n_r}$; (update the time of the known X_i)
    end for
  end for
end for

6.2.2 Non overlap LTS-DG schemes (NOLTS-DG)

Another way to explicitly estimate the value of $X\Big|_{\Gamma_i}$ appeared in [74] in finite difference context for heat equation. In which case there is no need of extending the boundary of the sub-domain Ω_i, once the local time steps Δt_i are defined. The key idea of the non overlap method, NOLTS-DG, is to first advance the solution globally to the time $t^n + \Delta t^*$ from the known solution at time t^n, where the global time step Δt^* larger than the maximum local time step Δt_{max}. This step is called the prediction step and is followed by an interpolation to obtain the values of $X\Big|_{\Gamma_i}$ needed to advance the solution locally on the sub-domain Ω_i. This last step is called the correction step. It has been applied in finite element context [73] and discontinuous Galerkin context [75] for parabolic equations.

In this section, we extend this approach to the DAREs in one, two or three spatial dimensions, using the DG method for the space discretization and time integrators such as Impl, ETD or EXPR for the resolution of the local system SO_i.

Non overlap LTS-DG algorithm

Once again, we consider the case where the solution domain Ω is split into three different sub-domains Ω_i with the local time step $\Delta t_i = 2^i \times \Delta t$ for all $i = 0, \cdots, m$, $m = 2$ and a given time step Δt. Therefore, in order to obtain the component X of the concentration entirely on Ω at the time t^{n+1}, from its known value at the time t^n using the non overlap LTS method, we use the following steps.

- **First step (Prediction):** advance the solution globally from t^n to $t^* = t^n + \Delta t^*$, by solving globally the DAREs using the DG spatial discretization method and a time integrator with uniform time step Δt^*. This is schematically illustrated in Fig 6.9.

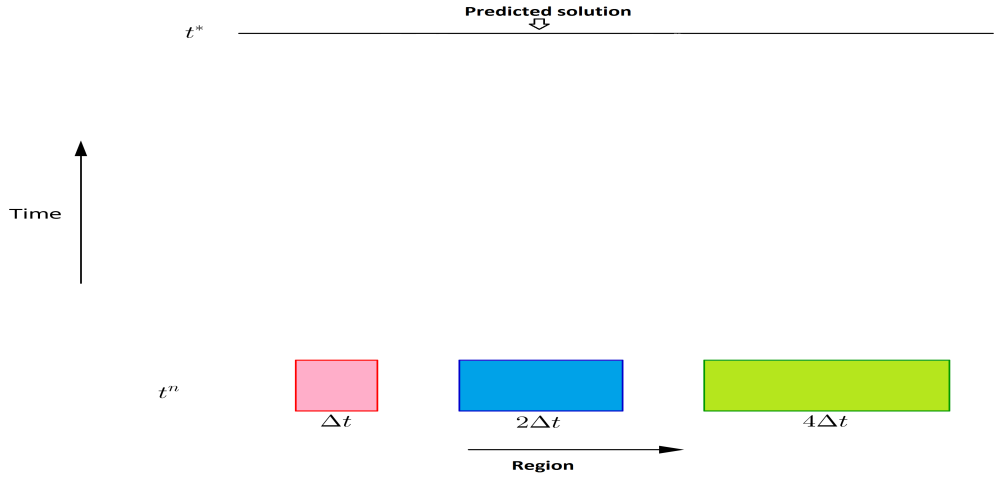


Figure 6.9: Prediction step of the non overlap LTS method, while the solution domain Ω is split into three regions Ω_0, Ω_1 and Ω_2 respectively with time step $\Delta t, 2\Delta t$ and $4\Delta t$.

- **Second step (Correction):** For all sub-domains Ω_i, use the known component X of the concentration at the time t^n and t^* to interpolate the value of Γ_i at at every time t_i^j, in order to advance the local component X_i from time t^n to t^{n+1}. This is schematically illustrated in Fig 6.10.

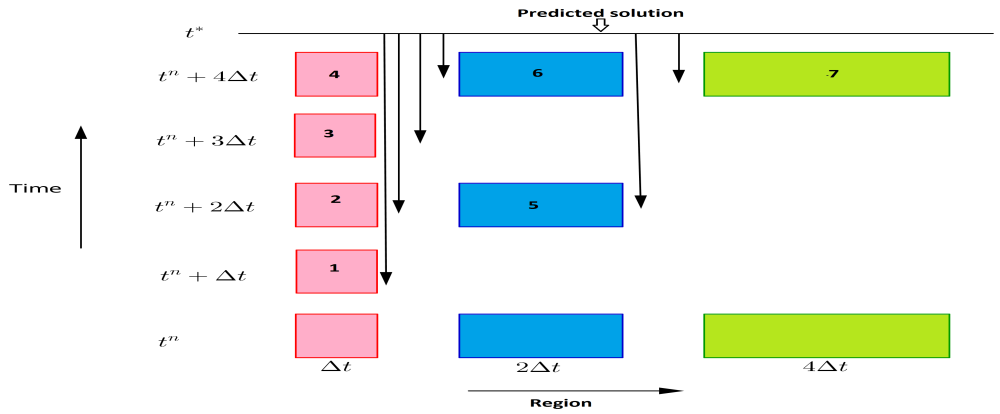


Figure 6.10: Correction step of the non overlap LTS method, while the solution domain Ω is split into three regions Ω_1, Ω_2 and Ω_3 respectively with time step $\Delta t, 2\Delta t$ and $4\Delta t$.

This process can be repeated, in order to estimate the solution at any final time T^1 from the known solution at initial time T^0. To that end, one can implement Algorithm 5 with $m_s = \frac{T^1 - T^0}{\Delta t_{\max}}$ and $r_i = \frac{\Delta t_{\max}}{\Delta t_i}$ for all $i \in \{0, \cdots, m\}$. Here, the function $\mathbf{I}$ is the interpolation function while the functions $\mathbf{A}_G$ and $\mathbf{A}_i$ are the iterative

function of the standard time integrators (described in Chapter 2) respectively used to solve the system of ODEs globally on Ω and locally on Ω_i.

Algorithm 5: Pseudo algorithm of the non overlap LTS method.

for $n = 0, \cdots, m_s - 1$; do (Advance the solution from T^0 to T^1)
  $t^n = T^0 + j \times \Delta t_{\max}$; (Computation of synchronized time t^n)
  $t^* = t^n + \Delta t^*$; (Computation of prediction time t^*)
  $X\Big|_{t^*} = \mathbf{A}_G \left(X\Big|_{t^n} \right)$;(Computation of predicted solution)
  parfor $i = 0, \cdots, m$; (Loop over all local regions - Correction step)
    $t^i_{old} = t^n$,
    for $j = 1, \cdots, r_i$; (Advance the solution on Ω_i from t^n to t^{n+1})
      $t^i_{new} = t^i_{old} + \Delta t_i$;
      $X\Big|_{\Gamma_i, t^i_{new}} = \mathbf{I} \left(X\Big|_{t^*}, X\Big|_{t^n} \right)$; (Estimate the needed boundary values)
      $X_i\Big|_{t^i_{new}} = \mathbf{A}_i \left(X_i\Big|_{t^i_{old}}, \Gamma_i\Big|_{t^i_{new}} \right)$; (Advance solution from t^i_{old} to t^i_{new})
      $t^i_{old} = t_{new}$;
    end for
  end parfor
end for

6.3 Numerical experiments

The goal of this section is to investigate the convergence of LTS-DG schemes and compare their efficiency against the one of the GTS-DG schemes while applied to several DAREs. Firstly in Section 6.3.1, by applying the OLTS-DG scheme to the two dimensional Ogata and Banks problem, described in Section 5.3.3, we examine how the direction of the bulk velocity of the DAREs and the the size of the overlap affect the accuracy of the OLTS-DG schemes. Secondly in Section 6.3.2, we compare the efficiency of the GTS-DG, OLTS-DG and NOLTS-DG schemes, by applying them to the one dimensional ETO model described in Section 3.2. Thirdly in Section 6.3.3, we examine the convergence and compare the efficiency of the GTS-DG and NOLTS-DG when applied to the transport of solute through a 2D domain with holes described in Section 5.3.4. Finally in Section 6.3.4, we examine the convergence and compare the efficiency of the GTS-DG and OLTS-DG when applied to the transport of solute through a 2D domain with fracture described in Section 5.3.5.

6.3.1 Effect of the bulk velocity and the size of overlap on the OLTS-DG schemes

The purpose of this section is to investigate how the direction of the bulk velocity or the size of the overlap and the order in which the solution restraints to the eligible sub-domains S^r are consecutively solved, affect the accuracy of the numerical solution obtained with OLTS-DG schemes. To that end, the $DG_{\text{OLTSD-Impl}}$ and $DG_{\text{OLTSI-Impl}}$ schemes are used to solve the Ogata Banks equation (described in Section 5.3.3) with the bulk velocity $\beta = (1, 0)$ and the diffusion coefficient $\epsilon = \frac{\|\beta\|}{Pe}$ where Pe is the Péclet number.

Effect of the bulk velocity on the OLTS-DG schemes

To investigate the effect of the bulk velocity on the accurracy of the OLTS-DG schemes, we conduct two numerical experiments. Here, we use the implicit Euler (Impl) scheme as the local time integrator. The solution domain $\Omega = [0, 1] \times [0, 1]$ is initially split into two sub-domains $\Omega_0 = [0, 0.4] \times [0, 1]$ and $\Omega_1 = [0.4, 1] \times [0, 1]$ with the local time steps Δt_0 and Δt_1, respectively. The overlapped sub-domains are defined as the union of the the initial sub-domains, Ω_i, and the neighbour elements of the mesh sharing node with the initial sub-domains Ω_i. We assume that the Péclet number is given by $Pe = 100$. We examine two cases: where Ω_0 has a fine and Ω_1 a coarse triangulation and then the reverse.

First, let us consider the region Ω_0 with the finer space triangulation and take $\Delta t_0 = 2^{-12}$ and $\Delta t_1 = 2^{-11}$. This is schematically shown on Fig 6.11, with the sequence in which the eligible sub-domains are considered for the solver $DG_{\text{OLTSD-Impl}}$ and $DG_{\text{OLTSI-Impl}}$. Note that the order in which $DG_{\text{OLTSI-Impl}}$ and $DG_{\text{OLTSD-Impl}}$ schemes update the local solution on eligible sub-domains respectively follow the direction and the opposite direction of the bulk velocity of the DAREs.

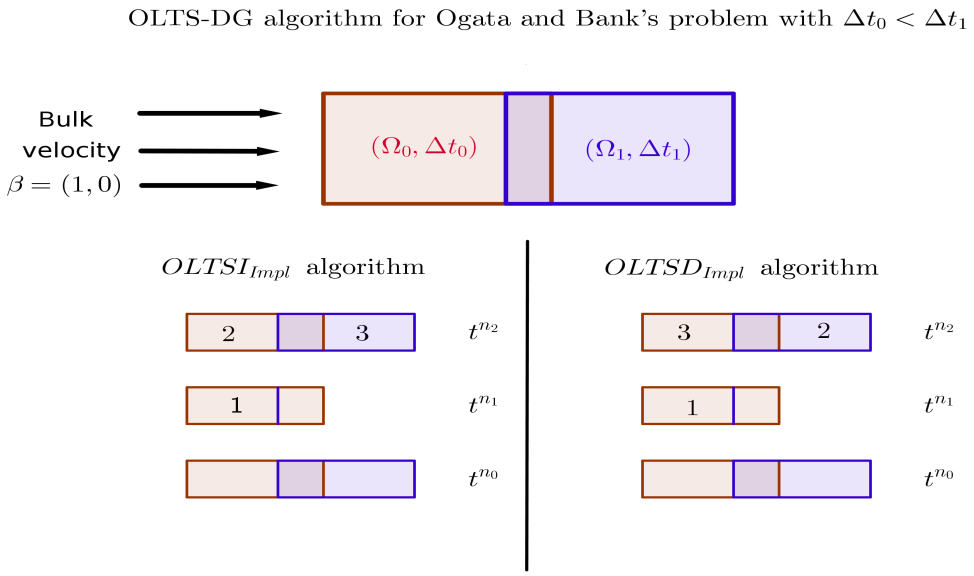


Figure 6.11: Domain decomposed such that the order, in which the local solution is updated in the case of $DG_{\text{OLTSI-Impl}}$ scheme, has the same direction as the bulk velocity. This is achieved by choosing $\Delta t_0 < \Delta t_1$ (e.g. $\Delta t_0 = 2^{-12}$ and $\Delta t_1 = 2^{-11}$). We also display the order in which the solution is updated in the case of $DG_{\text{OLTSI-Impl}}$ and $DG_{\text{OLTSD-Impl}}$ schemes.

We simulate the concentration C_r^i at the time $t = 0.25$ using the solvers $DG_{\text{OLTSD-Impl}}$ and $DG_{\text{OLTSI-Impl}}$ for every configuration of the time step $\Delta t_j^i = 4^i \times \Delta t_j$ for all $i \in \{0, 1, 2, 3\}$ and $j \in \{0, 1\}$. Note here that the local time step Δt_j^i decreases with the index i. For all the configurations $i \in \{0, 1, 2, 3\}$ and solvers DG_r, $r \in \{\text{OLTSD-Impl, OLTSI-Impl}\}$, we record the computational time, CPU_r^i, and compute the errors $error_r^i$ and E^i defined as follows

$$error_r^i = \left| C - C_r^i \right|_{L_2(\Omega)}, \quad E^i = \left| C_{OLTSD_{Impl}}^i - C_{OLTSI_{Impl}}^i \right|_{L_2(\Omega)} \tag{6.9}$$

where C is the exact concentration given by (5.108). In Tab 6.1, we display the values of $error_r^i$ and E_i. This shows that $DG_{\text{OLTSI-Impl}}$ is more accurate compared to $DG_{\text{OLTSD-Impl}}$. Here note that $DG_{\text{OLTSI-Impl}}$ updates the solution on the eligible sub-domains in the same direction as the bulk velocity. In Fig 6.12 (a) and (b), we plot the error $\log(error_r^i)$ respectively against $\log(\Delta t_1^i)$ and CPU_r^i. This shows that both overlap LTS-DG methods converge.

i	$error^i_{OLTSD_{Impl}}$	$error^i_{OLTSI_{Impl}}$	E^i
0	2.70×10^{-3}	2.64×10^{-3}	6.19×10^{-5}
1	4.29×10^{-3}	4.08×10^{-3}	2.60×10^{-4}
2	1.16×10^{-2}	1.08×10^{-2}	1.19×10^{-3}
3	3.60×10^{-2}	3.27×10^{-2}	6.19×10^{-3}

Table 6.1: We display in this table the values of $error^i_r$ and E_i, when the global domain is decomposed such that the order, in which the local solution is updated in the case of $DG_{\text{OLTSI-Impl}}$ scheme, has the same direction as the bulk velocity.

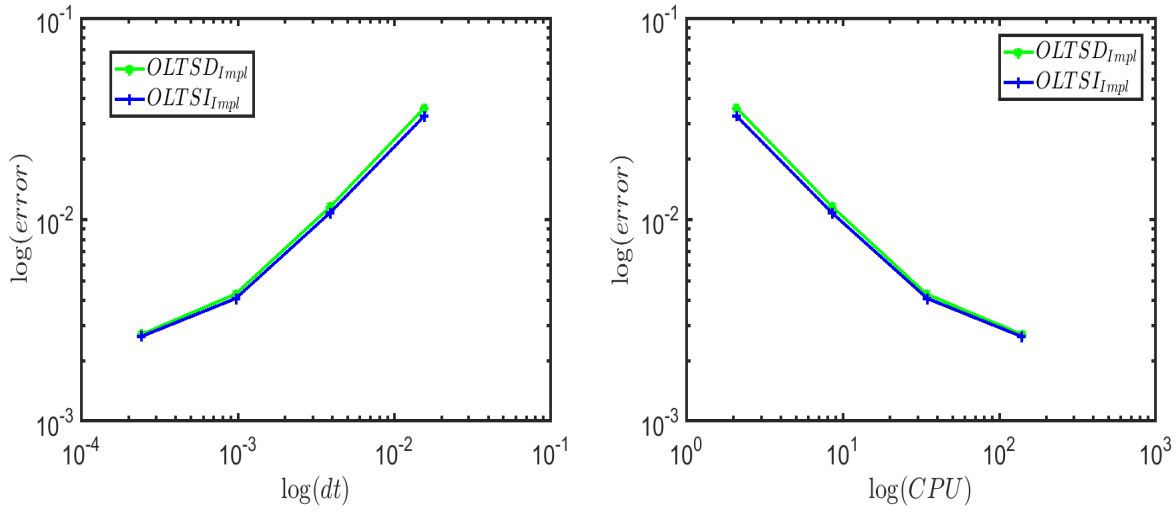


Figure 6.12: In (a) and (b), we plot the error $\log(error^i_r)$ respectively against $\log(dt = \Delta t^i_0)$ and CPU^i_r, when the global domain is decomposed such that the order, in which the local solution is updated in the case of $DG_{\text{OLTSI-Impl}}$ scheme, has the same direction as the bulk velocity.

Secondly, let us consider the region Ω_0 with the coarser space triangulation and takes $\Delta t_0 = 2^{-11}$ and $\Delta t_1 = 2^{-12}$. This is schematically shown on Fig 6.13, with the sequence in which the eligible sub-domains are considered for the solver $DG_{\text{OLTSD-Impl}}$ and $DG_{\text{OLTSI-Impl}}$. Note that the order in which $DG_{\text{OLTSD-Impl}}$ and $DG_{\text{OLTSI-Impl}}$ schemes update the local solution on eligible sub-domains respectively follow the direction and the opposite direction of the bulk velocity of the DAREs.

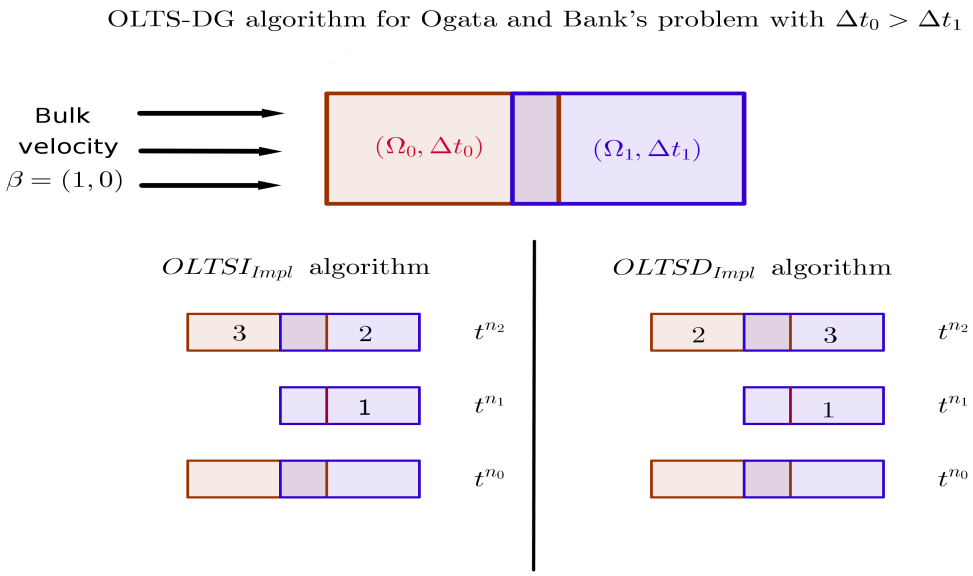
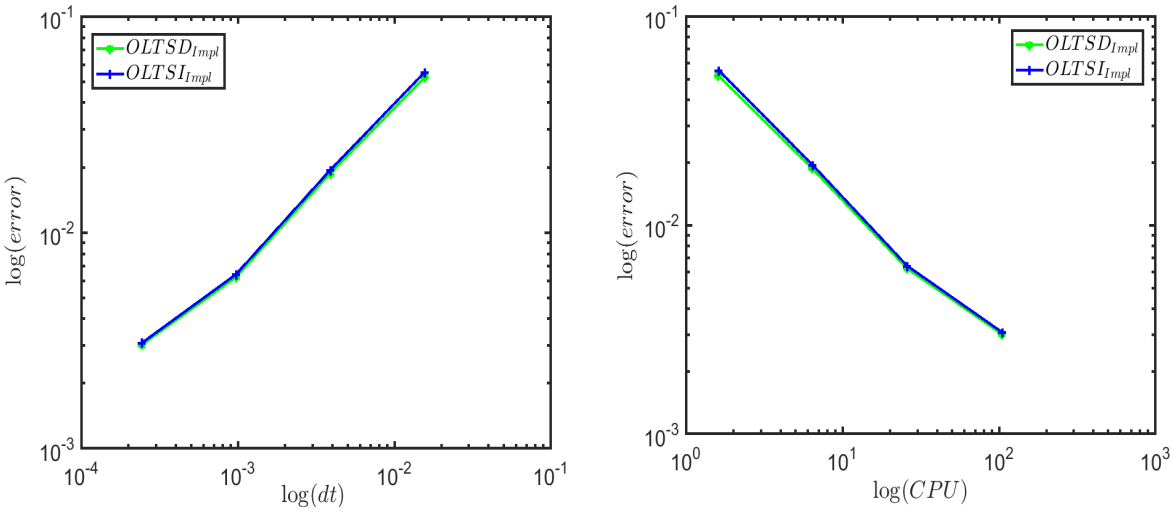


Figure 6.13: Domain decomposed such that the order in which the local solution is updated, in the case of $DG_{\text{OLTSI-Impl}}$ scheme, is in the opposite direction as the bulk velocity. This is achieved by choosing $\Delta t_0 > \Delta t_1$ (e.g. $\Delta t_0 = 2^{-11}$ and $\Delta t_1 = 2^{-12}$). Here, we also display the order in which the solution is updated in the case of $DG_{\text{OLTSI-Impl}}$ and $DG_{\text{OLTSD-Impl}}$ schemes.

We simulate the concentration C_r^i using the solvers $DG_{\text{OLTSD-Impl}}$ and $DG_{\text{OLTSI-Impl}}$ for every configuration of the time step $\Delta t_j^i = 4^i \times \Delta t_j$ for all $i \in \{0, 1, 2, 3\}$ and $j \in \{0, 1\}$. For all configurations $i \in \{0, 1, 2, 3\}$ and solvers DG_r, $r \in \{\text{OLTSD-Impl}, \text{OLTSI-Impl}\}$, we record the computational time, CPU_r^i, and compute the errors $error_r^i$ and E^i using (6.9). In Tab 6.2, we display the values of $error_r^i$ and E_i. This shows that $DG_{\text{OLTSD-Impl}}$ is more accurate compared to $DG_{\text{OLTSI-Impl}}$. Here $DG_{\text{OLTSD-Impl}}$ updates the solution on the eligible sub-domains in the same direction as the bulk velocity. In Fig 6.14 (a) and (b), we plot the error $\log(error_r^i)$ respectively against $\log(\Delta t_1^i)$ and CPU_r^i. This shows that both overlap LTS-DG methods converge.

i	$error^i_{OLTSD_{Impl}}$	$error^i_{OLTSI_{Impl}}$	E^i
0	3.02×10^{-3}	3.08×10^{-3}	6.27×10^{-5}
1	6.22×10^{-3}	6.41×10^{-3}	2.70×10^{-4}
2	1.88×10^{-2}	1.94×10^{-2}	1.31×10^{-3}
3	5.22×10^{-2}	5.51×10^{-2}	6.95×10^{-3}

Table 6.2: We display in this table the values of $error^i_r$ and E_i, when the global domain is decomposed such that the order, in which the local solution is updated in the case of $DG_{\text{OLTSI-Impl}}$ scheme, has the opposite direction as the bulk velocity.



(a) (b)

Figure 6.14: In (a) and (b), we plot the error $\log(error^i_r)$ respectively against $\log(dt = \Delta t^i_0)$ and CPU^i_r, when the global domain is decomposed such that the order, in which the local solution is updated in the case of $DG_{\text{OLTSI-Impl}}$ scheme, has the opposite direction as the bulk velocity.

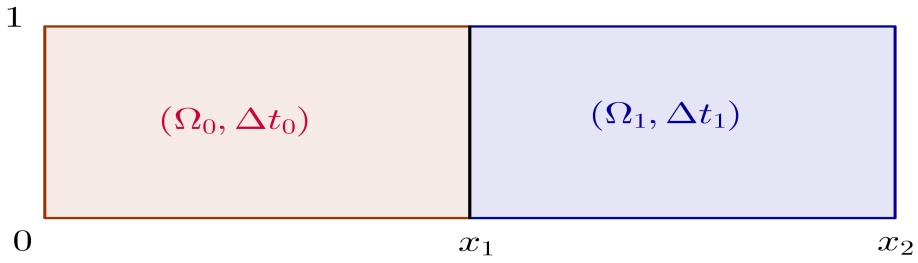
In the Ogata and Banks problem the fast change of the concentration of the solute takes place in a region close to the boundary at $x = 0$. So the sub-domain that contains the boundary at $x = 0$ should have the finest local time step, for a high accuracy of the OLTS-DG methods. This is illustrated by the better accuracy of both OLTS-DG methods obtained in the first experiment compared to the second experiment (comparing data in Tab 6.1 and Tab 6.2). Thus to improve the efficiency of the OLTS-DG method, the choice of the fine and coarse discretized sub-domains and the order of update of the local solution should respect the a priori physics.

Effect of the size of overlap on the OLTS-DG schemes

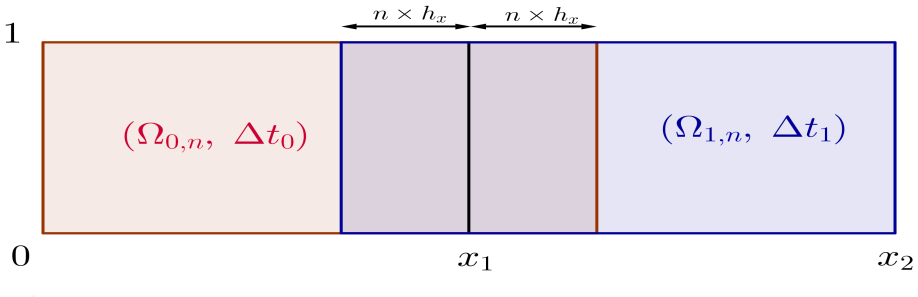
In this section, we investigate how the size of the overlap between two sub-domains affect the accuracy of the global solution. To that end, we consider the Ogata and Banks problem with the initial sub-domains $\Omega_0 = [0, x_1] \times [0, 1]$ and $\Omega_1 = [x_1, x_2] \times [0, 1]$, as illustrated in Fig 6.15 (a). For a given $n \in \{1, 2, \cdots, 11\}$ and $h_x = 0.02$, we consider the overlapped sub-domains $\Omega_{0,n}$ and $\Omega_{1,n}$ given by

$$\Omega_{0,n} = [0, x_1 + n \times h_x] \times [0, 1], \qquad \Omega_{1,n} = [x_1 - n \times h_x, x_2] \times [0, 1], \tag{6.10}$$

as illustrated in Fig 6.15 (b). Note that the size of the overlap (i.e. $(\Omega_{0,n} \bigcap \Omega_{1,n})$ is equal to $2n \times h_x$ and increases with n. The local time steps are $\Delta t_0 = 2^{-11}$ and $\Delta t_1 = 2^{-10}$, thus we consider the $DG_{\text{OLTSI-Impl}}$ scheme for more accuracy.



(a)



(b)

Figure 6.15: In (a) we shows the initial sub-domains $(\Omega_0, \Delta t_0)$ and $(\Omega_1, \Delta t_1)$. In (b), we illustrate the overlap sub-domains $(\Omega_{0,n},\ \Delta t_0)$ and $(\Omega_{1,n},\ \Delta t_1)$ for a given space step h_x and integer n. The size of the overlap is equal to $2n \times h_x$.

For all Péclet number $P_e \in \{0.1, 1, 10, 100\}$ and all $n \in \{1, 2, \cdots, 11\}$, we simulate the global solution, $C^n_{P_e}$ at the time $t = 0.5$, using the $DG_{\text{OLTSI-Impl}}$ scheme on the overlapped sub-domains $(\Omega_{0,n},\ \Delta t_0)$ and $(\Omega_{1,n},\ \Delta t_1)$. We then compute the

relative error $E^n_{P_e}$ as follows

$$E^n_{P_e} = \frac{\left|C^n_{P_e} - C\right|_{L^2(\Omega)}}{\left|C\right|_{L^2(\Omega)}}, \tag{6.11}$$

where C is the exact solution (given by 5.108) at the time $t = 0.5$ and Ω the solution domain. The results are summarized in Tab 6.3 and illustrated in Fig 6.16, where we plot the logarithm relative error ($\log(E^n_{P_e})$) against the integer n for all the Péclet numbers considered. A few conclusions can be drawn from the results in Tab 6.3 and Fig 6.16:

- For the Péclet number $P_e > 1$ ($\epsilon < 1$), the relative error $E^n_{P_e}$ increases slightly as we increase the size of the overlap (i.e. as we increase n). Thus, in the case of high Péclet number, the overlapped sub-domains obtained by including only the direct neighbour into the initial sub-domains, is the best choice to simulate efficiently the global solution.

- For the Péclet number $P_e \leq 1$ ($\epsilon \geq 1$), the relative error $E^n_{P_e}$ decreases as we increase the size of the overlap (i.e n).

- For large size of overlap, i.e. $n >> 1$, the relative error $E^n_{P_e}$ decreases as we decrease the Péclet number P_e (or increase ϵ). The same behaviour is observed in [114], when the overlapping Schwarz waveform relaxation scheme was applied to the viscous Burger equation with various values of the viscosity parameter. For the overlap equal 0.2 ($n = 10$ in our case), the error decreases as the diffusion term increases.

	Relative Error ($E^n_{P_e}$)			
	$P_e = 100, x_1 = 0.4, x_2 = 1$	$P_e = 10, x_1 = 0.8, x_2 = 2$	$P_e = 1, x_1 = 2, x_2 = 4$	$P_e = 0.1, x_1 = 7, x_2 = 14$
n = 1	8.2962×10^{-3}	6.8431×10^{-4}	9.5306×10^{-4}	1.3114×10^{-3}
n = 2	8.3353×10^{-3}	8.2657×10^{-4}	4.6384×10^{-4}	6.1631×10^{-4}
n= 3	8.3738×10^{-3}	8.8051×10^{-4}	3.1240×10^{-4}	4.0108×10^{-4}
n = 4	8.4275×10^{-3}	9.0963×10^{-4}	2.4465×10^{-4}	2.9913×10^{-4}
n = 5	8.4983×10^{-3}	9.2963×10^{-4}	2.0955×10^{-4}	2.4038×10^{-4}
n = 6	8.5827×10^{-3}	9.4555×10^{-4}	1.8985×10^{-4}	2.0268×10^{-4}
n = 7	8.6719×10^{-3}	9.5908×10^{-4}	1.7819×10^{-4}	1.7682×10^{-4}
n = 8	8.7528×10^{-3}	9.7075×10^{-4}	1.7101×10^{-4}	1.5829×10^{-4}
n = 9	8.8130×10^{-3}	9.8072×10^{-4}	1.6647×10^{-4}	1.4459×10^{-4}
n = 10	8.8442×10^{-3}	9.8910×10^{-4}	1.6355×10^{-4}	1.3422×10^{-4}
n = 11	8.8459×10^{-3}	9.9598×10^{-4}	1.6164×10^{-4}	1.2621×10^{-4}

Table 6.3: The relative error ($E^n_{P_e}$) for all Péclet number $P_e \in \{0.1, 1, 10, 100\}$ and all $n \in \{1, 2, \cdots, 11\}$. The size of the overlap is equal to $2n \times h_x$ with $h_x = 0.02$. Note that for large Péclet number P_e, ther is no advantage in taking large overlap.

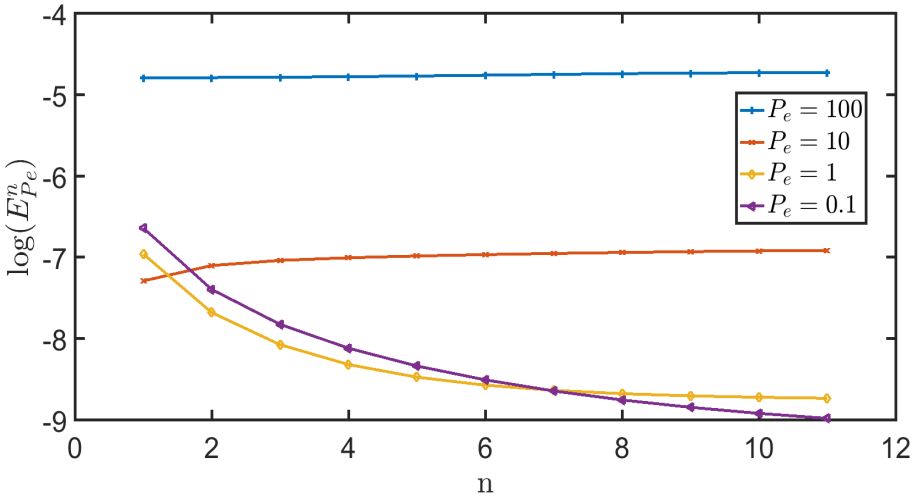


Figure 6.16: We plot the logarithm relative error ($\log(E^n_{P_e})$) against the integer n for all Péclet number considered. The size of the overlap is equal to $2n \times h_x$ with $h_x = 0.02$

6.3.2 Comparison of GTS-DG and LTS-DG schemes when applied to the 1D ETO model

In this section, we compare the performance of the GTS-DG and LTS-DG schemes, when solving the one dimension ETO model described in Chapter 3. Let us recall that GTS-DG schemes described in Chapter 3, unlike both LTS-DG schemes described in Section 6.2, use a time step defined uniformly on the domain solution. To that end, we first describe how we split the global solution domain. Secondly, we investigate the derivation of the local system SO_i associated to the sub-domain Ω_i for all $i = 0, \cdots, m$. Finally, we present the numerical results. According to the analysis presented in Chapter 3, the most efficient GTS-DG scheme was the one using the time integrator Impl. Therefore, we consider here the LTS-DG schemes that use the time integrator Impl as well. Thus in this case, we respectively denote $DG_{\text{OLTS-Impl}}$ and $DG_{\text{NOLTS-Impl}}$ the overlap and non overlap LTS-DG schemes.

As stated in Chapter 3, the solution domain is given by $\Omega = [0, z_{\max}]$, with $z_{\max}$ proportional to the diffusion length δ (i.e. $z_{\max} = k\delta,\ k \in \mathbb{N}$). Since the dimensionless current depends only on the concentration of the species at the boundary $z = 0$, we consider the partition $\Omega = \cup_{i=1}^{n} I_i$ where the interval $I_i = [z_{i-1}, z_i]$ is such that the step size $h_i = z_i - z_{i-1}$, respectively, follows the geometric and the uniform

progression on $[0, \delta]$ and $[\delta, z_{max}]$. Specifically for a given number $r \in \mathbb{N}$ and the increasing factor q, we have

$$h_i = \begin{cases} h \times q^{i-1}, & i = 1, \cdots, r \\ h \times q^{r}, & i = r+1, \cdots, n \end{cases}, h = \delta \left(\frac{q^r - 1}{q - 1} \right)^{-1}, n = r + \left\lceil \frac{z_{\max} - \delta}{h_r} \right\rceil.$$

Unless stated, for the simulation we use $q = 1.05$ and $r = 100$. We associate to each element I_i the time step $\Delta T_{I_i} \leq 2^{n_i}$, $n_i = \lceil \log_2 (\mathbf{C}_{max} h_i) \rceil$ for a given Courant number $\mathbf{C}_{max}$. We then update the time step on each element I_i by setting $\Delta T_{I_i} = \Delta T_{I_1}$ for all $i = 1, \cdots, r$ and $\Delta T_{I_i} = \Delta T_{I_{r+1}}$ for all $i = r+1, \cdots, n$.

For the simulation, we consider the geometry settings and the ETO model parameters given by

$$\delta = 20, z_{\max} = 5\delta, \mathbf{C}_{max} = 0.3, K_0 = 20, \tilde{D}^+ = 1. \tag{6.12}$$

This leads to the local time steps $\Delta t_{I_1} \leq 2^{-9}$ and $\Delta t_{I_{r+1}} \leq 2^{-2}$. We finally consider the sub-domains $\Omega_0 = [0, z_r]$ and $\Omega_1 = [z_r, z_n]$ with the local time step $\Delta t_0 = 2^{-9}$ and $\Delta t_1 = 2^{-6}$, respectively.

Local ODE system for the overlap LTS-DG scheme

In this section, we show how to extract the local ODE system for the OLTS-DG scheme from the global ODE system of the 1D ETO model, given by (3.53). To overlap the sub-domains here, we include the direct neighbour into the initial sub-domains. Thus, the overlapped sub-domains are $\Omega_0 = [0, z_{r+1}]$ and $\Omega_1 = [z_{r-1}, z_n]$, illustrated in Fig 6.17. In this case, we consider as internal boundary Γ_0 and Γ_1 as the node z_{r+1} of $[z_{r+1}, z_{r+2}]$ and z_{r-1} of $[z_{r-2}, z_{r-1}]$, respectively.

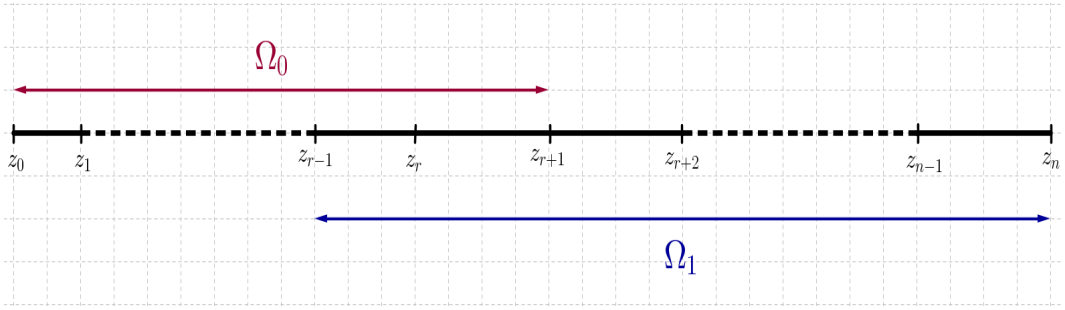


Figure 6.17: The overlapped regions for one dimension domain for ETO model.

The system of ODEs given by (3.53), obtained from the DG spatial discretization of the dimensionless governing equation of the ETO model, can be split into two systems of ODEs

$$\frac{d\chi_0}{d\tilde{t}} = L_0(\tilde{t})\chi_0 + \mathbb{S}_0^e\chi_1\Big|_{\Gamma_0}, \tag{6.13}$$

$$\frac{d\chi_1}{d\tilde{t}} = L_1(\tilde{t})\chi_1 + \mathbb{S}_1^e\chi_0\Big|_{\Gamma_1}, \tag{6.14}$$

where $\chi_j \in \mathbb{R}^{2n_{\Omega_j}}$ is the coupled component of the concentration of the species $\mathbf{Q}, \mathbf{Q}^+$ on the region Ω_j for all $j = 0, 1$. The dimensions n_{Ω_0} and n_{Ω_1} are given by

$$n_{\Omega_0} = \sum_{i=1}^{r+1}(k_i + 1), \qquad n_{\Omega_1} = \sum_{i=r}^{n}(k_i + 1), \tag{6.15}$$

where k_i is the highest degree of the Legendre polynomials considered on I_i (i.e. k_i+1 is the dimension of the DG finite space on I_i). The matrices $L_j, \mathbb{S}_j^e \in \mathbb{R}^{2n_{\Omega_j}\times 2n_{\Omega_j}}$ for all $j = 0, 1$ can be obtained from the matrix L given by (3.56) as follows

$$L_1 = \left(\begin{array}{c|c} DL_{0,0}^{s,\sigma_0} + K_f(\tilde{t})L^1 & -K_b(\tilde{t})L^1 \\ \hline -K_f(\tilde{t})L^1 & D^+L_{0,0}^{s,\sigma_0} + K_b(\tilde{t})L^1 \end{array}\right), \; L_2 = \left(\begin{array}{c|c} DL_{0,1}^{s,\sigma_0} & 0 \\ \hline 0 & D^+L_{0,1}^{s,\sigma_0} \end{array}\right). \tag{6.16}$$

Here, the matrix $L^1 \in \mathbb{R}^{n_{\Omega_0}\times n_{\Omega_0}}$ takes the same form as the matrix $L1$ defined in (3.46); the matrices $L_{0,0}^{s,\sigma_0}$ and $L_{0,1}^{s,\sigma_0}$ are efficiently extracted from the matrix L_0^{s,σ_0} (defined in (3.56)) as illustrated by Fig 6.18, due to its tridiagonalisation. Note from Fig 6.18 that only the block matrices $L_{0,I_{r+2}I_{r+1}}^{s,\sigma}$ and $L_{0,I_{r-1}I_r}^{s,\sigma}$, from the matrix L_0^{s,σ_0} respectively contribute to the computation of the vector $\mathbf{B}_0^T = \mathbb{S}_0^e\chi_1\Big|_{\Gamma_0}$ and

$\mathbf{B}_1^T = \mathbb{S}_1^e \chi_0\Big|_{\Gamma_1}$.

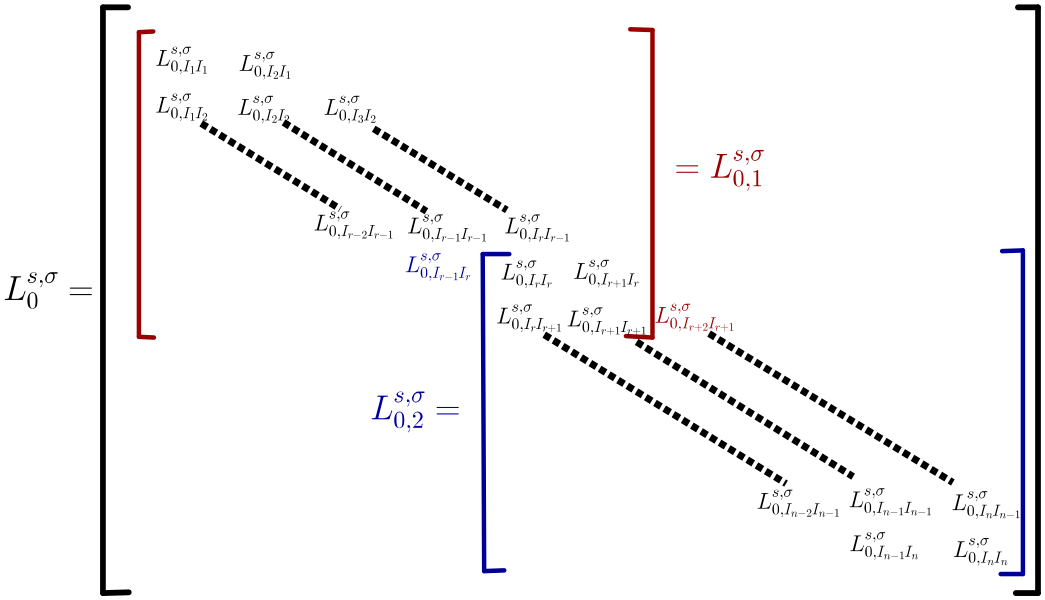


Figure 6.18: Extraction of the matrices $L_{0,1}^{s,\sigma_0}$ and $L_{0,1}^{s,\sigma_0}$ from the matrix L_0^{s,σ_0} when applying the overlap LTS method to the one dimension ETO model. It shows that only the block matrices $L_{0,I_{r+2}I_{r+1}}^{s,\sigma}$ and $L_{0,I_{r-1}I_r}^{s,\sigma}$, from the matrix L_0^{s,σ_0} respectively contribute to the computation of the vectors $\mathbf{B}_1^T$ and $\mathbf{B}_2^T$.

For all $j \in \{0,1\}$, we have $\mathbf{B}_j = [\mathbf{B}_j^{\alpha}, \mathbf{B}_j^{\alpha^+}]$ where the transpose of the block vectors $\mathbf{B}_j^{\alpha}, \mathbf{B}_j^{\alpha^+} \in \mathbb{R}^{1\times n_{\Omega_j}}$ are given by

$$\mathbf{B}_0^{\zeta} = \left(\mathbf{B}_{0,I_i}^{\zeta}\right)_{i=1,\cdots,r+1}, \quad \mathbf{B}_1^{\zeta} = \left(\mathbf{B}_{1,I_{p_i}}^{\zeta}\right)_{i=1,\cdots,n-r+1}, \; p_i = i + r - 1, \tag{6.17}$$

for all $\zeta = \alpha, \alpha^+$. According to the extraction of matrices illustrated in Fig 6.18, we have for all $\zeta = \alpha, \alpha^+$

$$\mathbf{B}_{0,I_i}^{\zeta} = \begin{cases} D^{\zeta} L_{0,I_{r+2}I_{r+1}}^{s,\sigma} \zeta_{I_{r+2}} & \text{if } \; i = r+1 \\ 0 & \text{if } \; i = 1,\cdots,r \end{cases},$$

$$\mathbf{B}_{1,I_{p_i}}^{\zeta} = \begin{cases} D^{\zeta} L_{0,I_{r-1}I_r}^{s,\sigma} \zeta_{I_{r-1}} & \text{if } \; i = 1 \\ 0 & \text{if } \; i = 2,\cdots,n-r+2 \end{cases},$$

where the coefficient D^{ζ} is the dimensionless diffusion coefficient of the species with the component in the DG finite space $\zeta = \alpha, \alpha^+$.

Local ODE system for the non overlap LTS-DG scheme

In this section, we show how to extract the local ODE system for the NOLTS-DG scheme from the global ODE system of the 1D ETO model, given by (3.53). The non overlapped sub-domains are $\Omega_0 = [0, z_r]$ and $\Omega_1 = [z_r, z_n]$, illustrated in Fig 6.25. In this case, we consider as internal boundary Γ_0 and Γ_1 as the node z_r of the interval $[z_r, z_{r+1}]$ and $[z_{r-1}, z_r]$, respectively.

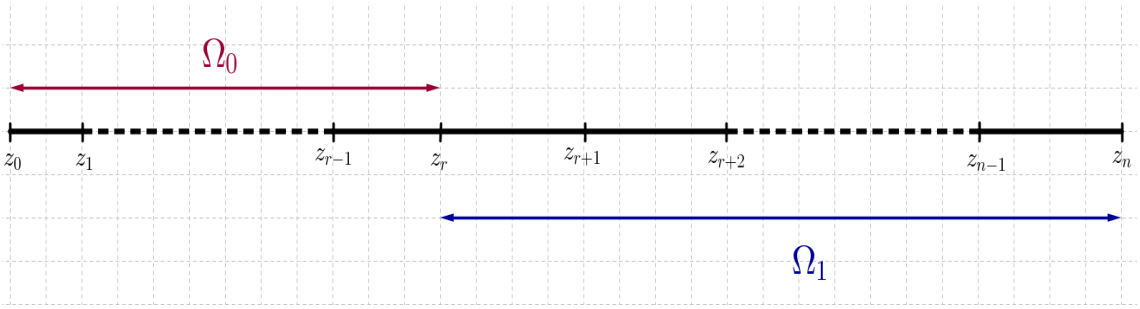


Figure 6.19: The non overlapped sub-domains for one dimension domain for ETO model.

The system of ODEs given by (3.53), obtained from the DG spatial discretization of the dimensionless governing equation of the ETO model, can be split into two ODEs system (6.13) and (6.14). In this case, the dimensions n_{Ω_0}, n_{Ω_1} are given by

$$n_{\Omega_0} = \sum_{i=1}^{r}(k_i + 1), \qquad n_{\Omega_1} = \sum_{i=r+1}^{n}(k_i + 1), \tag{6.18}$$

where k_i is the highest degree of the Legendre polynomials considered on I_i. Also, the matrices $L_j, \mathbb{S}_j^e \in \mathbb{R}^{2n_{\Omega_j} \times 2n_{\Omega_j}}$ for all $j = 0, 1$ are given by (6.16) where the matrix $L^1 \in \mathbb{R}^{n_{\Omega_0} \times n_{\Omega_0}}$ takes the same form as the matrix $L1$ defined in (3.46); the matrices $L_{0,0}^{s,\sigma_0}$ and $L_{0,1}^{s,\sigma_0}$ are efficiently extracted from the matrix L_0^{s,σ_0} (defined in (3.56)) as illustrated by Fig 6.20, due to its tridiagonalisation.

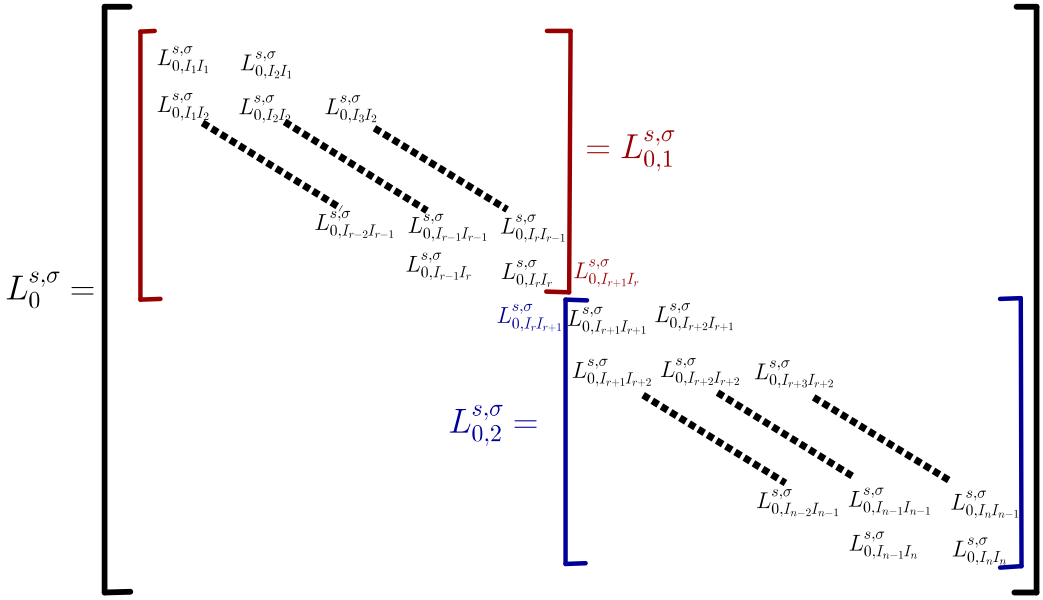


Figure 6.20: Extraction of the matrices $L_{0,1}^{s,\sigma_0}$ and $L_{0,1}^{s,\sigma_0}$ from the matrix L_0^{s,σ_0} when applying the non overlap LTS method to the one dimension ETO model. It shows that only the block matrices $L_{0,I_{r+1}I_r}^{s,\sigma}$ and $L_{0,I_rI_{r+1}}^{s,\sigma}$, from the matrix L_0^{s,σ_0} respectively contribute to the computation of the vectors $\mathbf{B}_1^T$ and $\mathbf{B}_2^T$.

Note from Fig 6.20 that only the block matrices $L_{0,I_{r+1}I_r}^{s,\sigma}$ and $L_{0,I_rI_{r+1}}^{s,\sigma}$, from the matrix L_0^{s,σ_0} respectively contribute to the computation of the vector $\mathbf{B}_0^T = \mathbb{S}_0^e \chi_1\Big|_{\Gamma_0}$ and $\mathbf{B}_1^T = \mathbb{S}_1^e \chi_0\Big|_{\Gamma_1}$. For all $j \in \{0,1\}$, we have $\mathbf{B}_j = [\mathbf{B}_j^\alpha, \mathbf{B}_j^{\alpha^+}]$ where the transposes of the block vectors $\mathbf{B}_j^\alpha, \mathbf{B}_j^{\alpha^+} \in \mathbb{R}^{1\times n_{\Omega_j}}$ are given by

$$\mathbf{B}_0^\zeta = \left(\mathbf{B}_{0,I_i}^\zeta\right)_{i=1,\cdots,r}, \quad \mathbf{B}_1^\zeta = \left(\mathbf{B}_{1,I_{p_i}}^\zeta\right)_{i=1,\cdots,n-r}, \; p_i = i + r. \tag{6.19}$$

According to the extraction of matrices illustrated in Fig 6.20, we have for all $\zeta = \alpha, \alpha^+$,

$$\mathbf{B}_{0,I_i}^\zeta = \begin{cases} D^\zeta L_{0,I_{r+1}I_r}^{s,\sigma} \zeta_{I_{r+1}} & \text{if } \; i = r \\ 0 & \text{if } \; i = 1,\cdots,r-1 \end{cases},$$

$$\mathbf{B}_{1,I_{p_i}}^\zeta = \begin{cases} D^\zeta L_{0,I_rI_{r+1}}^{s,\sigma} \zeta_{I_r} & \text{if } \; i = 1 \\ 0 & \text{if } \; i = 2,\cdots,n-r+1 \end{cases},$$

where the coefficient D^ζ is the dimensionless diffusion coefficient of the species with the component in the DG finite element space.

Numerical results of GTS-DG, OLTS-DG and NOLTS-DG schemes applied to the ETO model

Let us now focus on the numerical comparison of the accuracy and the efficiency of the GTS-DG and LTS-DG schemes, while simulating the dimensional current of the ETO model. To that end, for a given $i \in \{0, \cdots, 4\}$, we simulate the dimensionless current $G^{i,q}$, $q = DG_{\text{Impl}}, DG_{\text{OLTS-Impl}}, DG_{\text{NOLTS-Impl}}$ for the local time step $\Delta t_j^i = 2^{-i} \times \Delta t_j$ on the local solution domain Ω_j for all $j = 0, 1$. Note that for a given $i \in \{0, \cdots, 4\}$, the universal time step, $ht^i = \Delta t_0^i$, is considered for the GTS-DG schemes (i.e. the finest local time step of LTS-DG schemes). During the simulation of $G^{i,q}$, we also record the computation time $\text{CPU}^{i,q}$.

By assuming that the exact dimensionless current is given by $G^{4,DG_{\text{Impl}}}$, we then compare it against $G^{4,q}$ $q = DG_{\text{OLTS-Impl}}, DG_{\text{NOLTS-Impl}}$ to see which of the LTS-DG schemes is more accurate. This is illustrated in Fig 6.21, by plotting the dimensionless currents $G^{4,q}$ against the overpotential for all $q = DG_{\text{Impl}}, DG_{\text{OLTS-Impl}}, DG_{\text{NOLTS-Impl}}$ in Fig 6.21(a); and the absolute difference $E^q_{\text{abs}} = \left| G^{4,DG_{\text{Impl}}} - G^{4,q} \right|$ against the overpotential for all $q = DG_{\text{OLTS-Impl}}, DG_{\text{NOLTS-Impl}}$ in Fig 6.21(b).

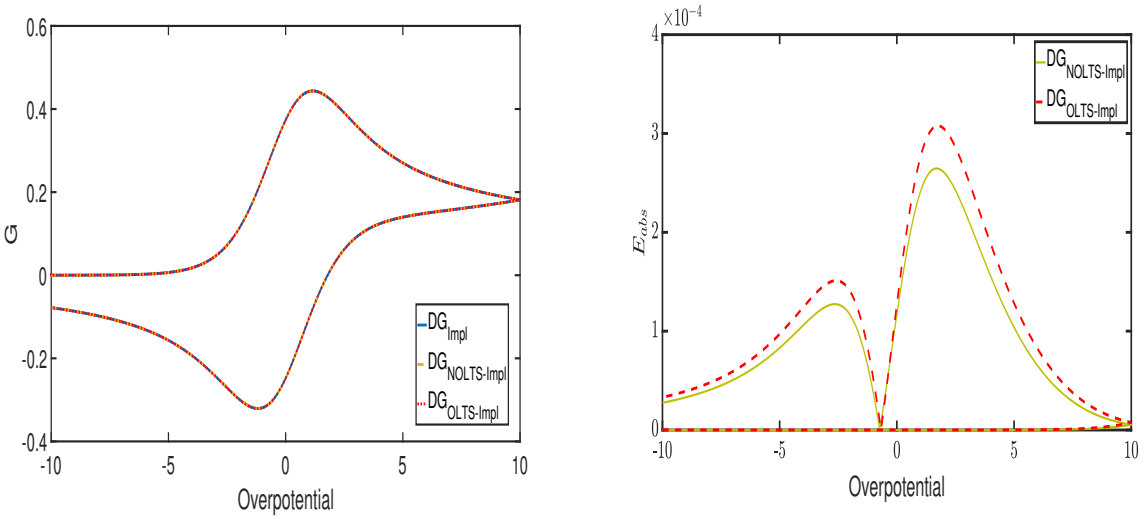


Figure 6.21: **Voltammogram of the ETO model simulate with GTS-DG and LTS-DG schemes.** In (a), we plot $G^{0,q}$ against the overpotential for all $q = DG_{\text{Impl}}, DG_{\text{OLTS-Impl}}, , DG_{\text{NOLTS-Impl}}$. In (b), we plot the absolute difference E^q_{abs} against the overpotential for all $q = DG_{\text{OLTS-Impl}}, , DG_{\text{NOLTS-Impl}}$.

To investigate the convergence and the efficiency of the GTS-DG and LTS-DG

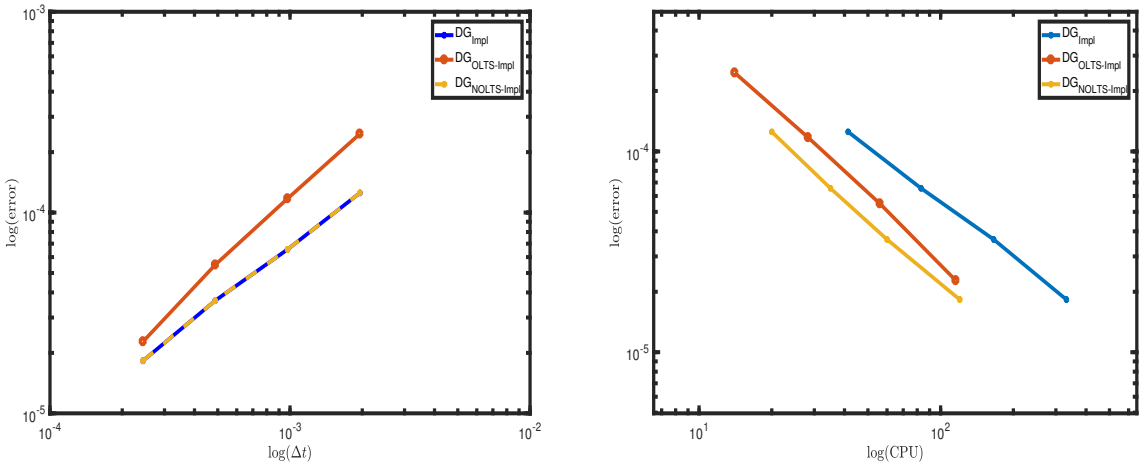
schemes, we compute the relative errors, error_q^i , given by

$$\text{error}_q^i = 100\frac{\left|G^{4,DG_{\text{Impl}}} - G^{i,q}\right|^2_{L^2(P)}}{\left|G^{4,DG_{\text{Impl}}}\right|^2_{L^2(P)}}, \tag{6.20}$$

for all $i = 0, \cdots, 3$ and all solver $q = DG_{\text{Impl}}, DG_{\text{OLTS-Impl}}, DG_{\text{NOLTS-Impl}}$. We then plot in Fig 6.22(a) the error , $\log(\text{error}_q^i)$, against the minimum of the local time step, $\log(ht^i)$. This shows the decay of the error with respect to the minimum local time step, meaning the DG_{Impl}, $DG_{\text{OLTS-Impl}}$, $DG_{\text{NOLTS-Impl}}$ converge in time. In Fig 6.22(b), we plot the error, $\log(\text{error}_q^i)$, against the computation time, $\log(\text{CPU}^{i,q})$. Note from Fig 6.22(b) that for a given error $E \in \mathbb{R}$ such that $E = \text{error}_q^i$, $q = DG_{\text{Impl}}, DG_{\text{OLTS-Impl}}, DG_{\text{NOLTS-Impl}}$, we have

$$\text{CPU}^{i,DG_{\text{NOLTS-Impl}}} < \text{CPU}^{i,DG_{\text{OLTS-Impl}}} < \text{CPU}^{i,DG_{\text{Impl}}}.$$

Fig 6.22(b) shows that the LTS-DG schemes, compared to GTS-DG schemes, are more efficient to simulate the dimensionless current of the ETO model.



(a) (b)

Figure 6.22: **Convergence and efficiency of the GTS-DG and LTS-DG schemes for the ETO model.** We respectively plot in (a) and (b) the error $\log(\text{error}_q^i)$ against the minimum time step $\log(ht^i)$ and the computation time $\log(\text{CPU}^{i,q})$ for all $i = 0, \cdots, 3$ and all $q = DG_{\text{Impl}}, DG_{\text{OLTS-Impl}}, DG_{\text{NOLTS-Impl}}$.

The 1D numerical experiment realized in this section has shown that

- even if the OLTS-DG schemes is not as accurate as GTS-DG scheme, the computation time of the OLTS-DG schemes is small enough to make them more efficient compare to the GTS-DG schemes,
- the non overlap LTS-DG scheme is more accurate than the overlap LTS-DG scheme. This is because the estimation of the needed internal value Γ_i, for the resolution of the system SO_i, is more accurate with the non overlap LTS-DG scheme.

6.3.3 Comparison of GTS-DG and NOLTS-DG schemes when applied to the 2D transport of solute through a domain with holes

The purpose of this section is to compare the accuracy and the efficiency of the GTS-DG and NOLTS-DG schemes while simulating transport of an inert solute within a fluid, with an absence of volumetric sources and sinks, through a 2D domain with holes. We consider here the GTS-DG scheme using the Impl as time integrator and NOLTS-DG schemes using the Impl, ETD1 as time integrators. Let us recall that, in Section 5.3.4, this problem has been investigated with the GTS-DG scheme using Impl as time integrator. During that investigation, we have

- described the construction of the refined mesh,
- simulated the velocity field from Darcy's equation,
- computated the local time step on every element of the triangulation for $\mathbf{C}_{max} = 0.8$, which split the solution domain, Ω into several sub-domains $(\Omega_i, \Delta t_i),\ i \in \{0, \cdots, m = 5\}$ schematically represented by the different colors in Fig 6.23,
- investigated the convergence and efficiency of GTS-DG scheme while simulating the concentration of the solute with the minimum local time step.

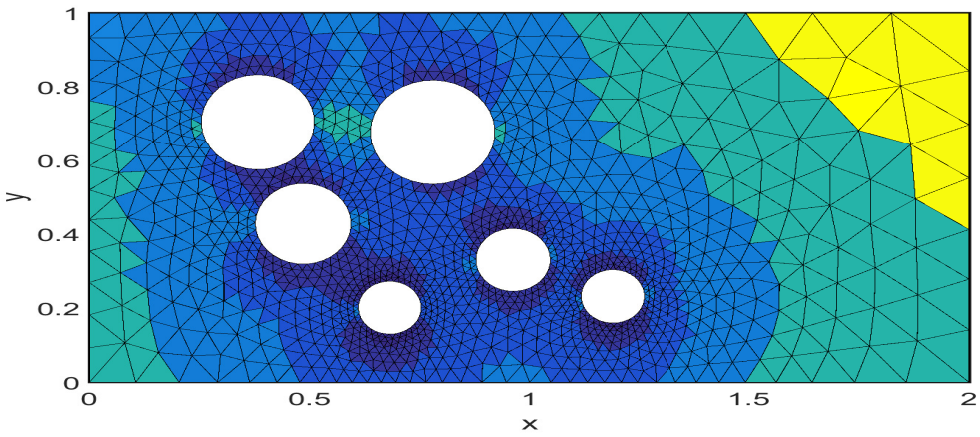


Figure 6.23: The non overlapped levels for two dimensions domain with holes for a given Courant number $\mathbf{C}_{max} = 0.08$. The colors represent the different sub-domains $\Omega_i,\ i = 0, \cdots, 5$ with different time steps.

Thus, to complete the goal of this section, we just need to investigate the problem under the same settings considered in Section 5.3.4 using the NOLTS-DG schemes.

Numerical results

For a given $r \in \{0, \cdots, 4\}$, we simulate the concentration C_p^r of the solute at the time $t = 1$ using the solver $p \in \{DG_{\text{NOLTS-Impl}}, DG_{\text{NOLTS-ETD1}}\}$ where the sub-domain Ω_i has the local time step $\Delta t_i^r = 2^{-r} \times \Delta t_i$ for all $i \in \{0, \cdots, 5\}$. Note that, in Section 5.3.4, we use the universal time step ht^r given by

$$ht^r = \min\{\Delta t_i^r,\ i = 0, \cdots, 5\},$$

to simulate the concentration $C_{DG_{\text{Impl}}}^r$ of the solute at the time $t = 1$ using the solver DG_{Impl} for a given $r \in \{0, \cdots, 4\}$. Throughout these simulations, we record the computation time,CPU_p^r , for all solver $p \in \{DG_{\text{Impl}}, DG_{\text{NOLTS-Impl}}, DG_{\text{NOLTS-ETD1}}\}$ and all $r \in \{0, \cdots, 4\}$. By assuming that the exact solution is given $C_{DG_{\text{Impl}}}^4$, we compute the absolute error, error_p^r given by

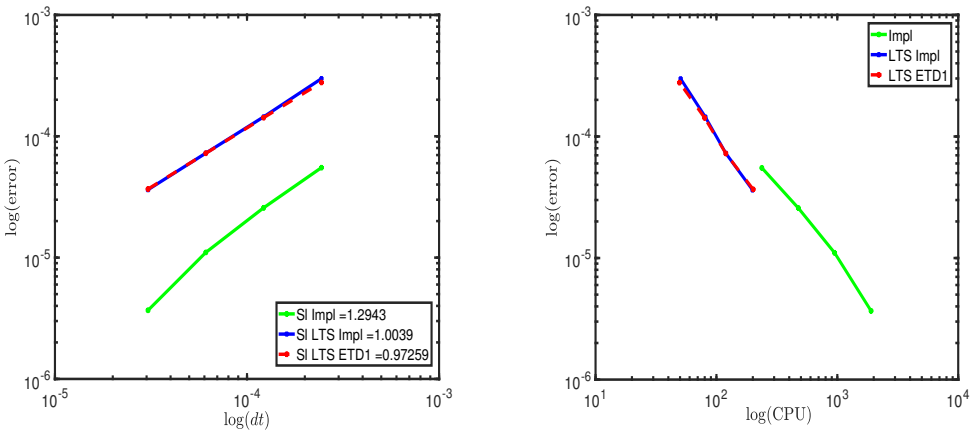
$$\text{error}_p^r = \| C_{DG_{\text{Impl}}}^4 - C_p^r \|_{L^2(\mathcal{T})},\quad r \in \{0, \cdots, 3\},$$

for all solvers $p \in \{DG_{\text{Impl}}, DG_{\text{NOLTS-Impl}}, DG_{\text{NOLTS-ETD1}}\}$. To investigate the convergence and efficiency of solvers considered here, we respectively plot in Fig 6.24(a)

and Fig 6.24(b) the error $\log(\text{error}_p^r)$ as a function of the minimum local time step $\log(ht^r)$ and the computation time $\log(\text{CPU}_p^r)$. Note from Fig 6.24(a) that the errors error_p^r decrease with the time step ht^r, meaning the GTS-DG and NOLTS-DG schemes, considered in this section, converge. Also, note from Fig 6.24(a) that

$$\text{error}_{DG_{\text{Impl}}}^r < \text{error}_p^r, \tag{6.21}$$

for all $p \in \{DG_{\text{NOLTS-Impl}}, DG_{\text{NOLTS-ETD1}}\}$. This shows that GTS-DG schemes is more accurate compared to NOLTS-DG schemes.This is expected since the GTS-DG schemes, unlike the NOLTS-DG schemes, consider the finest time step, uniformly on the solution domain. However, note from Fig 6.24(b) that the computation time is reduced enough to make the NOLTS-DG schemes more efficient compared to GTS-DG schemes.



(a) (b)

Figure 6.24: **Convergence and efficiency of the GTS-DG and NOLTS-DG schemes while solving the transport of solute through a domain with holes without presence of reaction.** We respectively plot in (a) and (b) the error $\log(\text{error}_p^r)$ as a function of the minimum local time step $\log(ht^r)$ and the computation time $\log(\text{CPU}_p^r)$. As expected, this shows that NOLTS-DG schemes is more efficient compared to GTS-DG schemes.

6.3.4 Comparison of GTS-DG and OLTS-DG schemes when applied to the 2D transport of solute through a domain with fracture

In this section, we compare the accuracy and the efficiency of the GTS-DG and OLTS-DG schemes while simulating transport of an inert solute within a fluid with or without an absence of volumetric sources and sinks through a 2D domain with fracture. In Section 5.3.5, these problems (i.e. reaction term given by $R(C) = 0$ or $R(C) = C - C^3$) were investigated with the GTS-DG scheme. Thus, we consider here the same problems under the same settings as in Section 5.3.5.

In Section 5.3.5, we described the construction of the refined mesh, simulated of the velocity field from Darcy's equation and computated the local time step Δt_T on every element T of the triangulation, which split the 2D domain with fracture into three sub-domains $(\Omega_i, \Delta t_i), i = 0, 1, 2$. Here, we consider the Péclet number $P_e = 3000$. So according to the analysis completed in Section 6.3.1, the suitable overlapped sub-domains are obtained by including the direct neighbour to the initial sub-domains. The overlapped sub-domains are illustrated in Fig 6.25 with the colors black, blue and yellow.

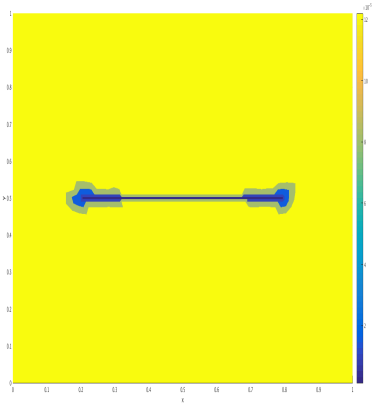


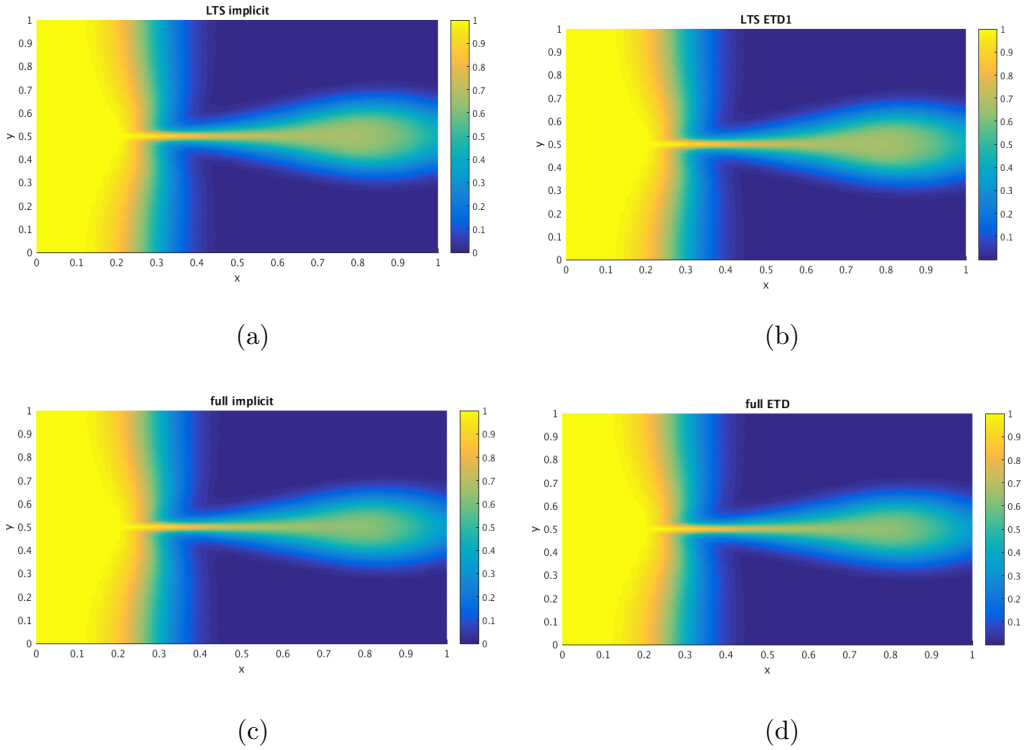
Figure 6.25: The overlapped sub-domains for two dimensional domain with fracture for a given Courant number $\mathbf{C}_{max} = 0.08$. The domain are represented by the color black, blue and yellow.

Numerical results for the case $R(C) = 0$

For a given $r \in \{0, \cdots, 4\}$, we simulate the concentration C_p^r of the solute at the time $t = 1$ using the solver $p \in \{DG_{\text{OLTS-Impl}}, DG_{\text{OLTS-ETD1}}\}$ where the sub-domain Ω_i has the local time step $\Delta t_i^r = 2^{-r} \times \Delta t_i$ for all $i \in \{0, \cdots, 5\}$. Note that, in Section 5.3.5, we use the universal time step ht^r given by

$$ht^r = \min\{\Delta t_i^r,\ i = 0, \cdots, 2\},$$

to simulate the concentration C_p^r of the solute at the time $t = 1$ using the solver $p \in \{DG_{\text{Impl}}, DG_{\text{ETD1}}\}$ for a given $r \in \{0, \cdots, 4\}$. Throughout these simulations (i.e. for all $r \in \{0, \cdots, 4\}$), we record the computation time, CPU_p^r, for all solver $p \in \{DG_q, DG_{\text{OLTS}-q}\}$ with $q \in \{\text{Impl, ETD1}\}$. We respectively plot in Fig 6.26 the concentration of the solute at the time $t = 1$, obtained with the solvers $DG_{\text{OLTS-Impl}}$, $DG_{\text{OLTS-ETD1}}$, DG_{Impl} and DG_{ETD1}.



(a) (b)

(c) (d)

Figure 6.26: **GTS-DG and OLTS-DG schemes applied to the transport of solute through a domain with fracture without presence of source and reaction.** We respectively plot in (a), (b), (c) and (d) the concentration of the solute at time $t = 1$, obtained with the solvers $DG_{OLTS-Impl}$, $DG_{OLTS-ETD1}$, DG_{Impl} and DG_{ETD1}. As expected, note the rapid transport and flow of the solute through the fracture.

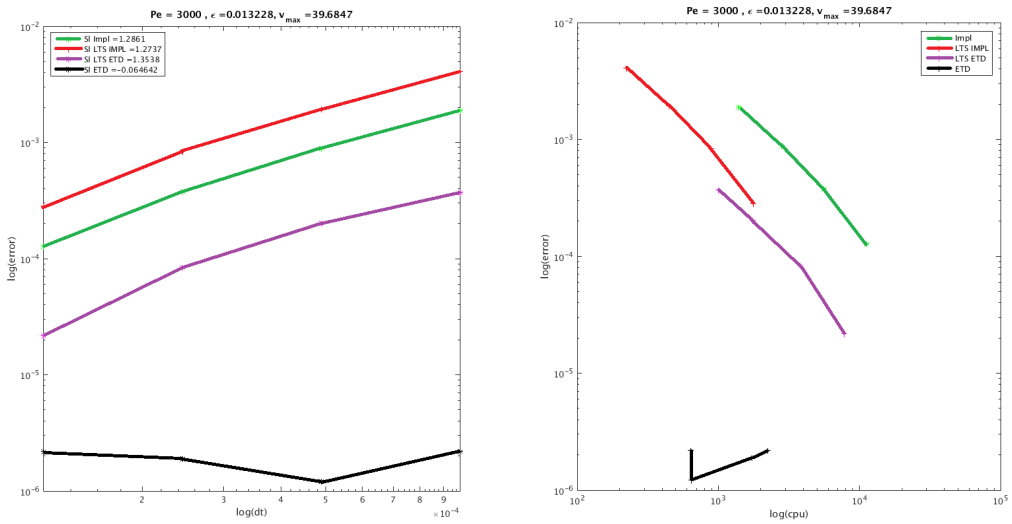
To investigate the convergence, the accuracy and efficiency of solvers $DG_{OLTS-Impl}$, $DG_{OLTS-ETD1}$, DG_{Impl} and DG_{ETD1}, we assume that for a given time integrator $q \in \{Impl, ETD1\}$, the exact concentration of the solute at the time $t = 1$ is given by $C^4_{DG_q}$. We then compute the error, error^r_p, given by

$$\text{error}^r_{DG_q} = \| C^4_{DG_q} - C^r_{DG_q} \|_{L^2(\mathcal{T})}, \quad \text{error}^r_{DG_{OLTS-q}} = \| C^4_{DG_q} - C^r_{DG_{OLTS-q}} \|_{L^2(\mathcal{T})},$$

for all $r \in \{0, \cdots, 3\}$ and all time integrator $q \in \{Impl, ETD1\}$. We plot in Fig 6.27(a) the errors $\log(\text{error}^r_{DG_q})$ and $\log(\text{error}^r_{DG_{OLTS-q}})$ against $\log(ht^r)$ for all $r = 0, \cdots, 3$ and $q =$ Impl, ETD1. We plot in Fig 6.27(b) the errors $\log(\text{error}^r_{DG_q})$ and $\log(\text{error}^r_{DG_{OLTS-q}})$ against $\log(\text{CPU}^r_{DG_q})$ and $\log(\text{CPU}^r_{DG_{OLTS-q}})$ for all $r = 0, \cdots, 3$ and $q =$ Impl, ETD1. Note from Fig 6.27(a) that the errors error^r_p decrease with the time step ht^r, meaning the GTS-DG and OLTS-DG schemes, considered in this section, converge. Also, note from Fig 6.27(a) that

$$\text{error}^r_{DG_q} < \text{error}^r_{DG_{OLTS-q}}, \tag{6.22}$$

for all time integrator $q \in \{Impl, ETD1\}$. This shows that the OLTS-DG schemes is less accurate compared to GTS-DG schemes. This is expected since the GTS-DG schemes, unlike the OLTS-DG schemes, consider the finest time step, uniformly on the solution domain. However, note from Fig 6.27(b) that the computation time is reduced enough to make the OLTS-DG schemes more efficient compared to GTS-DG schemes.



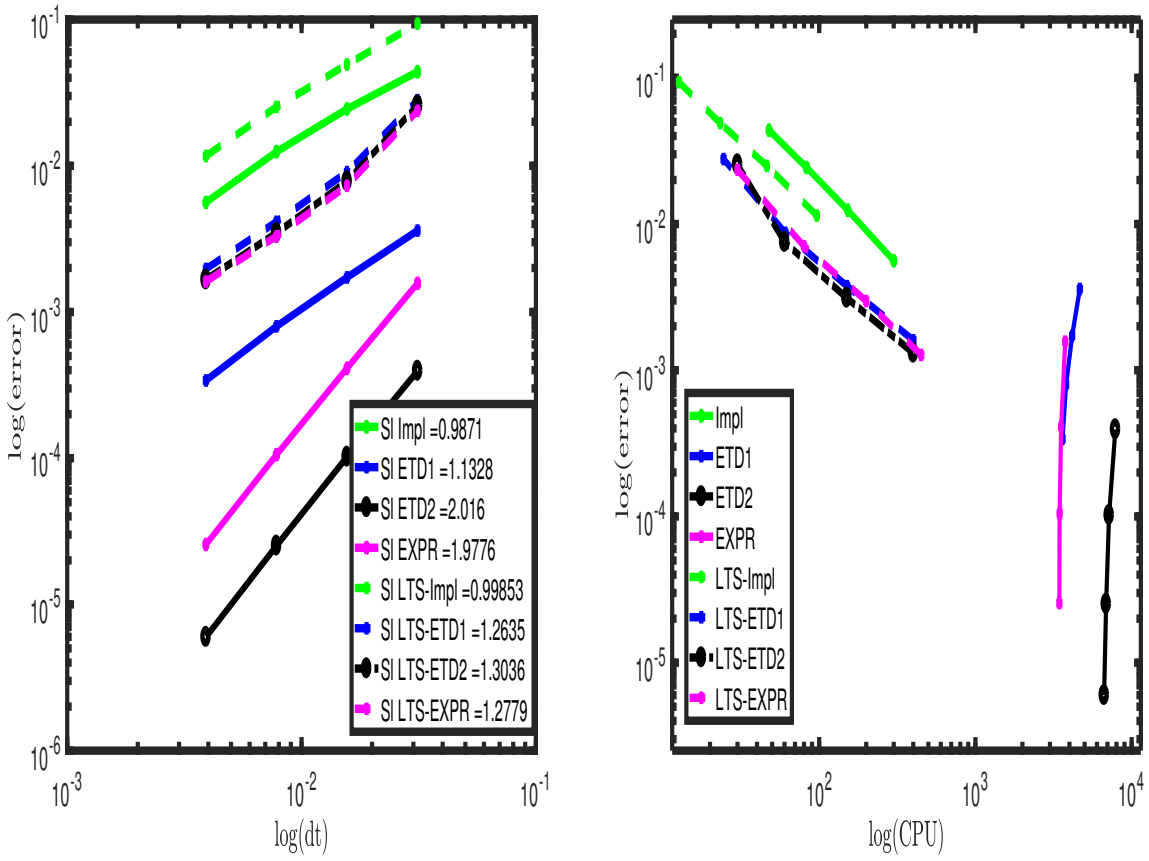
(a) (b)

Figure 6.27: **Convergence and efficiency of GTS-DG and OLTS-DG schemes while solving the transport of solute through a domain with fracture without presence of source and reaction.** In (a), we plot $\log(\text{error}^r_{DG_q})$ and $\log(\text{error}^r_{DG_{OLTS-q}})$ against $\log(ht^r)$ for all $r = 0, \cdots, 3$ and $q =$ Impl, ETD1. In (b), we respectively plot $\log(\text{error}^r_{DG_q})$ and $\log(\text{error}^r_{DG_{OLTS-q}})$ against $\log(\text{CPU}^r_{DG_q})$ and $\log(\text{CPU}^r_{DG_{OLTS-q}})$ for all $r = 0, \cdots, 3$ and $q =$ Impl, ETD1.

Numerical results for the case $R(C) = C - C^3$

Let us consider the flow and transport of solute through a domain with fracture with the presence of non linear reaction term, given by $R(C) = C^3 - C$. We have investigated this problem in Section 5.3.5, with the GTS-DG solvers DG_q, $q \in \{\text{Impl, ETD1, ETD2, EXPR}\}$. By considering the same settings, used in Section 5.3.5, we compare the accuracy and efficiency of the solvers DG_q and DG_{OLTS-q} with $q \in \{\text{Impl, ETD1, ETD2, EXPR}\}$. We use the previous strategy to illustrate the results of this comparison in Fig 6.28.

In Fig 6.28 (a), we plot $\log(\text{error}^r_{DG_q})$ and $\log(\text{error}^r_{DG_{OLTS-q}})$ against $\log(ht^r)$ for all $r = 0, \cdots, 3$ and $q \in \{\text{Impl, ETD1, ETD2, EXPR}\}$. In Fig 6.28 (b), we respectively plot $\log(\text{error}^r_{DG_q})$ and $\log(\text{error}^r_{DG_{OLTS-q}})$ against $\log(\text{CPU}^r_{DG_q})$ and $\log(\text{CPU}^r_{DG_{OLTS-q}})$ for all $r = 0, \cdots, 3$ and $q \in \{\text{Impl, ETD1, ETD2, EXPR}\}$.



(a) (b)

Figure 6.28: **Convergence and efficiency of GTS-DG and OLTS-DG schemes while solving the transport of solute through a domain with fracture with presence of reaction.** In (a), we plot $\log(\text{error}^r_{DG_q})$ and $\log(\text{error}^r_{DG_{OLTS-q}})$ against $\log(ht^r)$ for all $r = 0, \cdots, 3$ and $q =$ Impl, ETD1, ETD2, EXPR. In (b), we respectively plot $\log(\text{error}^r_{DG_q})$ and $\log(\text{error}^r_{DG_{OLTS-q}})$ against $\log(\text{CPU}^r_{DG_q})$ and $\log(\text{CPU}^r_{DG_{OLTS-q}})$ for all $r = 0, \cdots, 3$ and $q \in$ {Impl, ETD1, ETD2, EXPR}.

Note from Fig 6.28 (a) that the errors error^r_p decrease with the time step ht^r, meaning the GTS-DG and OLTS-DG schemes, considered in this section, converge. Also, note from Fig 6.28(a) that

$$\text{error}^r_{DG_q} < \text{error}^r_{DG_{OLTS-q}}, \tag{6.23}$$

for all time integrator $q \in$ {Impl, ETD1, ETD2, EXPR}. This shows that OLTS-DG schemes is less accurate compared to GTS-DG schemes. This is expected since the GTS-DG schemes, unlike the OLTS-DG schemes, consider the finest time step, uniformly on the solution domain. However, note from Fig 6.28 (b) that the com-

putation time is reduced enough to make the OLTS-DG schemes more efficient compared to GTS-DG schemes. As stated in Section 5.3.5, the GTS-DG with time integrator ETD1, ETD2 and EXPR are almost constant due to the computation of time of $e^L X$ using *phipm*.

6.4 Summary

In order to efficiently capture the localized small-scale physics of DAREs on a complex geometry, we developed here two solvers, the overlap and non overlap LTS-DG schemes, based on the domain decomposition techniques, the DG spatial discretization method and the standard time integrators such as Impl, ETD, EXPR presented in Chapter 2. The several numerical investigations lead to the following findings:

- When applied to Ogata and Banks problem, the numerical results of the overlap LTS-DG method show that the choice of the fast and slow components can significantly affect the accuracy of the solution. A better accuracy is obtained if the eligible sub-domains are considered in the same direction as the bulk velocity. In a high Péclet number regime, unlike in the low Péclet number regime, the size of the overlap doesn't improve the accuracy of the overlap LTS-DG method.

- When applied to the one dimension ETO model and the two dimension transport of solute through a domain with fracture or holes, the numerical results showed that the computation time is reduced enough to make the LTS-DG schemes proposed here more efficient compared to the GTS-DG schemes. These numerical results also showed that the non overlap LTS-DG method is more accurate and efficient compared to the overlap LTS-DG method. This is due to the fact that the needed information are more accurately computed in the case of the non overlap LTS method.

Chapter 7

Conclusion and Future Work

This chapter recapitulates the key results, main contributions and proposes recommendations for future research.

The primary aim of this thesis is the development of new efficient numerical methods to solve DAREs, specifically to investigate the one dimension cyclic volatmmetry models (ETO and EC' models) and the two dimension flow and transport of solute in porous media. To that end, we focus on the method of lines in which the evolution problem is first discretized in space using the FE or DG method, yielding a system of coupled ODEs. The obtained system of ODEs is then discretized in time using globally or locally the time solvers such as Impl, ETD, EXPR or ROCK2.

In Chapter 2, we review the time integrators schemes (Impl, ETD, EXPR and ROCK2) considered in this thesis. We also investigate numerically the performance of Impl, ETD, EXPR and ROCK2 when used to solve DAREs. To that end, we consider the method of lines with FE method as the spatial discretization method.

From Chapter 3 to Chapter 4, we develop a numerical inversion method for the one dimensional cyclic voltammetry models by fitting the entire signal response (i.e. the dimensionless current G). The inversion problem is equivalent to a PDEs-constrained minimization problem defined as follows

$$\underset{p\in\mathfrak{P}\subset\mathbb{R}^n}{\arg\min}\left\{F(p)=\int_0^{s_t}\frac{1}{2}(G^p-G^{obs})^2dt \quad s.t. \quad G^p=W(p)\right\}, \tag{7.1}$$

where s_t is the duration of the scan, G^{obs} is the measured signal response and G^p is

the signal response obtained by solving the PDEs, W, for a given parameter p. We numerically show that we can efficiently compute

- the function G^p, thus $F(p)$, by solving the PDEs, W, with the method of lines which combines the DG method and the time integrator Impl (see Chapter 3),
- the gradient $\nabla F(p)$ by using the adjoint method (AD) (see Chapter 4)

for a given parameter p and measured signal response G^{obs}. Regardless of the dimension n of the parameter p, the computation of the functions $F(p)$ and $\nabla F(p)$, using DG, Impl and AD methods, is then reduced to the resolution of the PDEs, W, and an ODE called the adjoint equation. In order to estimate the solution of (7.1), the functions $F(p)$ and $\nabla F(p)$ (obtained with DG and AD methods) are supplied to the gradient descent algorithm such as `fmincon` of MATLAB.

The performance of the proposed numerical inversion method is then compared to the one that used MATLAB's `pdepe` (which is the combination of the classic FE method with the time solver $ode15s$) to only supply $F(p)$ to `fmincon`. This investigation has shown that the better conservation property of the DG method compared to the classic FE method improves enormously the efficiency of the simulation and the fitting of the voltammogram. For the ETO model, unlike the EC' model, the novel numerical method computes efficiently the parameter model, and leads to better estimation for both models when compared to the MATLAB's `pdepe` based code.

Note that the results presented in Chapter 3 and Chapter 4 represent a part of the mathematical aspects of the work completed during my internship at Schlumberger Gould Research (SGR) centre in Cambridge. Previously the approach involving `pdepe` was used for the inversion of the signal response and was limited by the unacceptable computation time and inefficient fitting. These limitations were removed by combining DG, Impl and AD methods.

Chapter 5 is devoted to the approximation of the model of flow and transport in porous media (in 2 or 3 dimensions), using the IP-DG method together with a time integrator. To do so, we revisit the IP-DG schemes reviewed by Ern in [80], to semidiscretize the governing equation (DAREs) and assemble the matrices that

arise from the semidiscretization (i.e. mass and stiffness matrices). We compare the performance of the IP-DG method combined with the time solvers described in Section 2.3, to simulate the flow and transport of solute through a two dimensional domain with fracture or holes). The numerical experiments have shown that this approach is accurate, however it can become greatly expensive in term of computational time.

In Chapter 6, we use the prior knowledge of the solution domain to construct two LTS methods (overlap and non overlap methods) in order to reduce the computational time of the numerical resolution of the DAREs. The LTS method combines the domain decomposition techniques and the standard time step method used locally on several broken regions of the solution domain with different time steps. We then investigate the one dimensional cyclic voltammetry models and the two dimensions transport and flow in porous media with the combination of the proposed LTS methods and the DG method. These investigations have shown that in general, when combined with the DG method, the LTS methods are more efficient compared to the standard time step methods used globally with the same time step. But more specifically, numerical experiment in one dimension indicates that the non overlap method is more efficient than the overlap method.

7.1 Future Work

The results mentioned above, motivate further research:

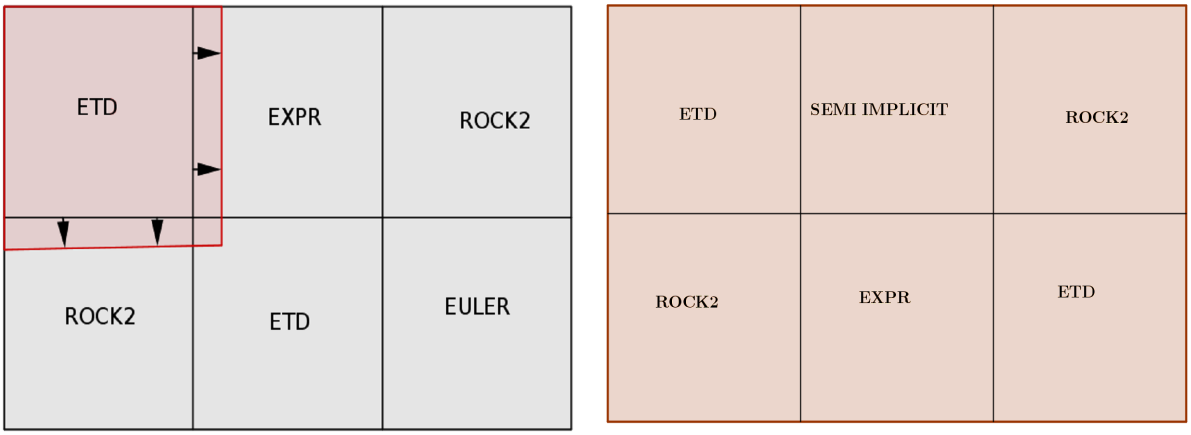
- Improve the the numerical inversion method for the EC' model by using for example the Tikhonov regularisation method [197, 171, 93, 122] or another optimisation algorithm (stochastic or deterministic).

- Combine the DG method, LTS methods, the adjoint method and the MATLAB code `fmincon` to develop a new numerical inversion method for the cyclic voltammetry models.

- Carry out a theoretical analysis to prove that the convergence of the numerical methods based on the combination of the DG and LTS methods proposed in

this thesis is independent of the number of subdomains.

- Introduce a new LTS method that will use locally the more efficient time solver as illustrated in Fig 7.1.



(a) LTS overlap (b) LTS Non overlap

Figure 7.1: **LTS method with different local time solvers.**

- Extend the application of the proposed numerical methods to the stochastic partial differential equations.

Appendix A

Appendix

In this appendix, we include the full details of some computations needed for the construction and the validation of the numerical inversion method for the simulation and the inversion of the one dimension cyclic voltammetry models. We also include a code in MATHEMATICA that help for the symbolic computations.

A.1 Some properties of Legendre polynomials

In this section, we focus on the computation of several quantities needed for the assemble of the mass and stiffness matrices that arise from the DG semi discretization, based on the Legendre polynomials, of the cyclic voltammetry models. Let $\{P_r,\ \ r \geq 0\}$ be the set of Legendre polynomials of degree less or equal to k on the interval $I = [-1, 1]$, then we have

$$P_r(1) = 1, \qquad P_r(-1) = (-1)^r, \qquad d_x P_l(1) = \frac{l(l+1)}{2}. \tag{A.1}$$

$$\int_I P_r(x) = 2\delta_r^0, \quad \int_I P_r(x) P_l(x) dx = \frac{2\delta_r^l}{2r+1}. \tag{A.2}$$

The derivative $d_x P_r$ is given by the linear combination of Legendre polynomials as follows

$$d_x P_l(x) = \frac{2P_{(l-1)}(x)}{\| P_{(l-1)}(x) \|^2} + \frac{2P_{(l-1)-2}(x)}{\| P_{(l-1)-2}(x) \|^2} + \frac{2P_{(l-1)-4}(x)}{\| P_{(l-1)-4}(x) \|^2} + \cdots \tag{A.3}$$

Therefore we have

$$\begin{aligned} d_x P_l(-1) =& \frac{2P_{(l-1)}(-1)}{\| P_{(l-1)}(x) \|^2} + \frac{2P_{(l-1)-2}(-1)}{\| P_{(l-1)-2}(x) \|^2} + \frac{2P_{(l-1)-4}(-1)}{\| P_{(l-1)-4}(x) \|^2} + \cdots \\ =& \frac{2(-1)^{(l-1)}}{\| P_{(l-1)}(x) \|^2} + \frac{2(-1)^{(l-1)-2}}{\| P_{(l-1)-2}(x) \|^2} + \frac{2(-1)^{(l-1)-4}}{\| P_{(l-1)-4}(x) \|^2} + \cdots \\ =& (-1)^{(l-1)} \left(\frac{2}{\| P_{(l-1)}(x) \|^2} + \frac{2}{\| P_{(l-1)-2}(x) \|^2} + \frac{2}{\| P_{(l-1)-4}(x) \|^2} + \cdots \right) \\ =& (-1)^{(l-1)} d_x P_l(1) \\ d_x P_l(-1) =& \frac{l(l+1)}{2} (-1)^{(l-1)}. \end{aligned} \tag{A.4}$$

Combining (A.1) and (A.3), with the orthogonality of the Legendre polynomials, we obtain the following results

- if $r + l$ is odd then

$$\int_I d_x P_r(x) d_x P_l(x) dx = 0 \tag{A.5}$$

- if $r + l$ is even and $l \leq r$ then

$$\begin{aligned} \int_I d_x P_r(x) d_x P_l(x) dx =& \| d_x P_l(x) \|^2 \\ =& \frac{4}{\| P_{(l-1)}(x) \|^2} + \frac{4}{\| P_{(l-1)-2}(x) \|^2} + \frac{4}{\| P_{(l-1)-4}(x) \|^2} + \cdots \\ =& 2 d_x P_l(1) \\ \int_I d_x P_r(x) d_x P_l(x) dx =& l(l+1) \end{aligned} \tag{A.6}$$

A.2 Projection in the DG finite space V_h

The goal in this section is to approximate the initial condition of the cyclic voltammetry models by a function in the DG finite space V_h (given by (3.38)), in order to simulate the voltammogram. Since the DG finite space V_h is orthonormal, then for a given function $F(z)$, we have

$$F(z) \approx \sum_{j=1}^{n} \sum_{r=0}^{k_j} a_r^j \Phi_r^j(z), \quad \text{with} \quad a_r^j = \int_\Omega F(z) \Phi_r^j(z) \, dz. \tag{A.7}$$

If $F(z) = C^0, C^0 \in \mathbb{R}$, then by combining (3.37) and (A.2), we have

$$\begin{aligned} a_r^j =& C^0 \int_{I_j} \frac{\phi_r^j(z)}{\parallel \phi_r^j \parallel_{L^2(I_j)}} \, dz \\ =& \frac{C^0 h_j}{2 \parallel \phi_r^j \parallel_{L^2(I_j)}} \int_I P_r(x) \, dx \\ =& \frac{C^0 h_j \delta_r^0}{\parallel \phi_r^j \parallel_{L^2(I_j)}} \\ a_r^j =& \begin{cases} C^0 \sqrt{h_j} & \text{if } \ r = 0, \forall j \in \{1, \cdots, n\} \\ 0, & \text{otherwise} \end{cases} . \end{aligned} \tag{A.8}$$

A.3 The local stiffness matrix stemming from the interior nodes

The purpose of this section is to give the details of the computation of the entries of the local stiffness matrix stemming from the interior nodes, summarized in (3.51). For a given the index of the interior nodes $i \in \{1, \cdots, n-1\}$ and basis functions Φ_r^{r1}, Φ_l^{r2}, we have

$$\mathcal{C}_i^{s,\sigma_0}(\Phi_r^{r1}, \Phi_l^{r2}) = \Gamma_{\{r\}}^{r1} \Gamma_{\{l\}}^{r2} \mathcal{C}_i^{s,\sigma_0}(\phi_r^{r1}, \phi_l^{r2}),$$

with $0 \leq r \leq k_{r1}, \ 0 \leq l \leq k_{r2}, \quad r1, r2 = i, i+1$. By combining the properties of the Legendre polynomials stated in Section A.1 and the definition the bilinear form $\mathcal{C}_i^{s,\sigma_0}$ given by (3.32), we have the following results

$$\begin{aligned} \mathcal{C}_i^{s,\sigma_0}(\phi_r^i, \phi_l^i) =& - \{d_z \phi_r^i(z_i)\}[\phi_l^i(z_i)] + s\{d_z \phi_l^i(z_i)\}[\phi_r^i(z_i)] + \sigma_i [\phi_l^i(z_i)][\phi_r^i(z_i)] \\ =& - \frac{1}{2} d_z \phi_r^i(z_i^-) \phi_l^i(z_i^-) + s \frac{1}{2} d_z \phi_l^i(z_i^-) \phi_r^i(z_i^-) + \sigma_i \phi_l^i(z_i^-) \phi_r^i(z_i^-) \\ =& - \frac{1}{h_i} d_x P_r(1) P_l(1) + s \frac{1}{h_i} d_x P_l(1) P_r(1) + \sigma_i P_l(1) P_r(1) \\ \mathcal{C}_i^{s,\sigma_0}(\phi_r^i, \phi_l^i) =& - \frac{r(r+1)}{2h_i} + s \frac{l(l+1)}{2h_i} + \sigma_i \end{aligned}$$

$$
\begin{aligned}
\mathcal{C}_i^{s,\sigma_0}(\phi_r^i,\phi_l^{i+1}) =& -\{d_z\phi_r^i(z_i)\}[\phi_l^{i+1}(z_i)] + s\{d_z\phi_l^{i+1}(z_i)\}[\phi_r^i(z_i)] + \sigma_i[\phi_r^i(z_i)][\phi_l^{i+1}(z_i)] \\
=& \frac{1}{2}d_z\phi_r^i(z_i^-)\phi_l^{i+1}(z_i^+) + s\frac{1}{2}d_z\phi_l^{i+1}(z_i^+)\phi_r^i(z_i^-) \\
& - \sigma_i\phi_r^i(z_i^-)\phi_l^{i+1}(z_i^+) \\
=& \frac{1}{h_i}d_xP_r(1)P_l(-1) + s\frac{1}{h_{i+1}}d_xP_l(-1)P_r(1) - \sigma_iP_r(1)P_l(-1) \\
=& \left(\frac{r(r+1)}{2h_i}(-1)^l + s\frac{l(l+1)}{2h_{i+1}}(-1)^{l-1} - \sigma_i(-1)^l\right) \\
\mathcal{C}_i^{s,\sigma_0}(\phi_r^i,\phi_l^{i+1}) =& \left(\frac{r(r+1)}{2h_i} - s\frac{l(l+1)}{2h_{i+1}} - \sigma_i\right)(-1)^l
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{C}_i^{s,\sigma_0}(\phi_r^{i+1},\phi_l^i) =& -\{d_z\phi_r^{i+1}(z_i)\}[\phi_l^i(z_i)] + s\{d_z\phi_l^i(z_i)\}[\phi_r^{i+1}(z_i)] + \sigma_i[\phi_l^i(z_i)][\phi_r^{i+1}(z_i)] \\
=& -\frac{1}{2}d_z\phi_r^{i+1}(z_i^+)\phi_l^i(z_i^-) - s\frac{1}{2}d_z\phi_l^i(z_i^-)\}\phi_r^{i+1}(z_i^+) \\
& - \sigma_i\phi_l^i(z_i^-)\phi_r^{i+1}(z_i^+) \\
=& -\frac{1}{h_{i+1}}d_xP_r(-1)P_l(1) - s\frac{1}{h_i}d_xP_l(1)\}P_r(-1) - \sigma_iP_l(1)P_r(-1) \\
=& \left(-\frac{r(r+1)}{2h_{i+1}}(-1)^{r-1} - s\frac{l(l+1)}{2h_i}(-1)^r - \sigma_i(-1)^r\right) \\
\mathcal{C}_i^{s,\sigma_0}(\phi_r^{i+1},\phi_l^i) =& \left(\frac{r(r+1)}{2h_{i+1}} - s\frac{l(l+1)}{2h_i} - \sigma_i\right)(-1)^r
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{C}_i^{s,\sigma_0}(\phi_r^{i+1},\phi_l^{i+1}) =& -\{d_z\phi_r^{i+1}(z_i)\}[\phi_l^{i+1}(z_i)] + s\{d_z\phi_l^{i+1}(z_i)\}[\phi_r^{i+1}(z_i)] + \sigma_i[\phi_l^{i+1}(z_i)][\phi_r^{i+1}(z_i)] \\
=& \frac{1}{2}d_z\phi_r^{i+1}(z_i^+)\phi_l^{i+1}(z_i^+) - s\frac{1}{2}d_z\phi_l^{i+1}(z_i^+)\phi_r^{i+1}(z_i^+) \\
& + \sigma_i\phi_l^{i+1}(z_i^+)\phi_r^{i+1}(z_i^+) \\
=& \frac{1}{h_{i+1}}d_xP_r(-1)P_l(-1) - s\frac{1}{h_{i+1}}d_xP_l(-1)P_r(-1) \\
& + \sigma_iP_l(-1)P_r(-1) \\
=& \frac{1}{h_{i+1}}\frac{r(r+1)}{2}(-1)^{r-1}(-1)^l - s\frac{1}{h_{i+1}}\frac{l(l+1)}{2}(-1)^{l-1}(-1)^r \\
& + \sigma_i(-1)^l(-1)^r \\
\mathcal{C}_i^{s,\sigma_0}(\phi_r^{i+1},\phi_l^{i+1}) =& \left(-\frac{r(r+1)}{2h_{i+1}} + s\frac{l(l+1)}{2h_{i+1}} + \sigma_i\right)(-1)^{l+r}
\end{aligned}
$$

A.4 Computation and update of the matrices N_1, J_1 and J_2

We explain here how to efficiently assembly and update, the vector N_1 and the matrices J_1, J_2, needed for the numerical resolution of the model EC'. We have shown in Section 3.3.3 that N_1, J_1 and J_2 take the form

$$N_1 = (N_1^j)^T_{j\in\{1,\cdots,n\}}, \quad J_1 = \mathrm{Diag}(J_1^j)_{j\in\{1,\cdots,n\}}, \quad J_2 = \mathrm{Diag}(J_2^j)_{j\in\{1,\cdots,n\}},$$

where N_1^j is a (k_j+1) block vector with entries given by (3.90), J_1^j and J_2^j are a $(k_j+1)\times(k_j+1)$ block matrices with entries given by (3.94). The block matrices J_1^j, J_2^j and the block vector N_1^j depend on $\alpha^{+,j}$ and $\beta^{-,j}$ (respectively the component of the species $\mathbf{Q}^+$ and B on the element I_j). For the seek of clarity, we use the notation

$$\alpha_l^{+,j} = a_l, \quad \beta_l^{-,j} = b_l, \quad \forall l \in \{0,\cdots,k_j\}.$$

A.4.1 Computation of the matrices N_1^j, J_1^j and J_2^j

We run Algorithm 6 in Mathematica, in order to compute the matrices N_1^j, J_1^j and J_2^j for all $k_j \in \{1,2,3\}$ and a given components of the species $\mathbf{Q}^+$, B on the interval I_j (respectively denoted a_l, b_l). The result is summarised in Tab A.1 and Tab A.2. The expressions of the block vector N_1^j are presented in Tab A.1 and the expressions of the block matrices J_1^j, J_1^j are presented in Tab A.2. Tab A.2 shows that the block matrices J_1^j depends on $b_l, l \in \{1,\cdots,k_j\}$ while the block matrices J_2^j depends on $a_l, l \in \{1,\cdots,k_j\}$.

Algorithm 6: Computation of the matrices N_1^j, J_1^j and J_2^j for all $k_j \in \{1, 2, 3\}$.

```
(*===Code for the computation of the integral of triple product===*)
tripLeg[i0_, m0_, q0_] :=Module[{i = i0, m = m0, q = q0}],
If[Abs[m − i] ≤ q&&q ≤ m + i,
z = 2 ∗ (ThreeJSymbol[{i, 0}, {m, 0}, {q, 0}])^2; z = 0]; z ]
(∗ ===== Code for the computation of the source term N_1^j ===== ∗)
SourceTermOrthonormal[k1_, h_, A1_, B1_]:=
Module[{k = k1, hj = h, A2 = A1, B2 = B1},
For[l = 0; M = IdentityMatrix[k + 1]; ST = Array[0&, k + 1], l ≤ k, + + l,
For[r = 0, r ≤ k, + + r, For[n = 0; n ≤ k, + + n,
M[[r + 1; n + 1]] = (Sqrt[(2 ∗ l + 1) ∗ (2 ∗ r + 1) ∗ (2 ∗ n + 1)]) ∗ tripLeg[r, n, l]]];
ST[[l + 1]] = A2.M.B2]; (1/(2 ∗ Sqrt[hj])) ∗ ST]
"− − − − − − − − − Computation of N_1^j, J_1^j and J_2^j for k_j = 1 − − − − − − − − − "
k = 1
A = {a_0, a_1}
B = {b_0, b_1}
MatrixForm[SourceTermOrthonormal[k, hj, A, B]]
MatrixForm[D[SourceTermOrthonormal[k, hj, A, B], {A}]]
MatrixForm[D[SourceTermOrthonormal[k, hj, A, B], {B}]]
MatrixForm[D[SourceTermOrthonormal[k, hj, A1, B1], {B1}]]
"− − − − − − − − − Computation of N_1^j, J_1^j and J_2^j for k_j = 2 − − − − − − − − − "
k = 2
A = {a_0, a_1, a_2}
B = {b_0, b_1, b_2}
MatrixForm[SourceTermOrthonormal[k, hj, A, B]]
MatrixForm[D[SourceTermOrthonormal[k, hj, A, B], {A}]]
MatrixForm[D[SourceTermOrthonormal[k, hj, A, B], {B}]]
"− − − − − − − −Computation of N_1^j, J_1^j and J_2^j for k_j = 3 − − − − − − − − − "
k = 3
A = {a_0, a_1, a_2, a_3}
B = {b_0, b_1, b_2, b_3}
MatrixForm[SourceTermOrthonormal[k, hj, A, B]]
MatrixForm[D[SourceTermOrthonormal[k, hj, A, B], {A}]]
MatrixForm[D[SourceTermOrthonormal[k, hj, A, B], {B}]]
```

k_j	$N_1^{j^T}$
1	$\frac{1}{2\sqrt{h_j}} \begin{pmatrix} 2a_0b_0 + 2a_1b_1 \\ 2a_1b_0 + 2a_0b_1 \end{pmatrix}$
2	$\frac{1}{2\sqrt{h_j}} \begin{pmatrix} 2a_0b_0 + 2a_1b_1 + 2a_2b_2 \\ 2a_1b_0 + \left(2a_0 + \frac{4}{\sqrt{5}}a_2\right) b_1 + \frac{4}{\sqrt{5}}a_1b_2 \\ 2a_2b_0 + \frac{4}{\sqrt{5}}a_1b_1 + \left(2a_0 + \frac{4\sqrt{5}}{7}a_2\right) b_2 \end{pmatrix}$
3	$\frac{1}{2\sqrt{h_j}} \begin{pmatrix} 2a_0b_0 + 2a_1b_1 + 2a_2b_2 + 2a_3b_3 \\ 2a_1b_0 + \left(2a_0 + \frac{4}{\sqrt{5}}a_2\right) b_1 + \left(\frac{4}{\sqrt{5}}a_1 + 6\sqrt{\frac{3}{35}}a_3\right) b_2 + 6\sqrt{\frac{3}{35}}a_2b_3 \\ 2a_2b_0 + \left(\frac{4}{\sqrt{5}}a_1 + 6\sqrt{\frac{3}{35}}a_3\right) b_1 + \left(2a_0 + \frac{4\sqrt{5}}{7}a_2\right) b_2 + \left(6\sqrt{\frac{3}{35}}a_1 + \frac{8}{3\sqrt{5}}a_3\right) b_3 \\ 2a_3b_0 + 6\sqrt{\frac{3}{35}}a_2b_1 + \left(6\sqrt{\frac{3}{35}}a_1 + \frac{8}{3\sqrt{5}}a_3\right) b_2 + \left(2a_0 + \frac{8}{3\sqrt{5}}a_2\right) b_3 \end{pmatrix}$

Table A.1: **Computation of the block vector N_1^j :** This table gives the expression of the block vector N_1^j for all $k_j \in \{1, 2, 3\}$ and a given components of the species $\mathbf{Q}^+$, B on the interval I_j (respectively denoted a_l, b_l).

k_j	J_1^j	J_2^j
1	$\frac{1}{2\sqrt{h_j}}\begin{pmatrix} 2b_0 & 2b_1 \\ 2b_1 & 2b_0 \end{pmatrix}$	$\frac{1}{2\sqrt{h_j}}\begin{pmatrix} 2a_0 & 2a_1 \\ 2a_1 & 2a_0 \end{pmatrix}$
2	$\frac{1}{2\sqrt{h_j}}\begin{pmatrix} 2b_0 & 2b_1 & 2b_2 \\ 2b_1 & 2b_0+\frac{4}{\sqrt{5}}b_2 & \frac{4}{\sqrt{5}}b_1 \\ 2b_2 & \frac{4}{\sqrt{5}}b_1 & 2b_0+\frac{4\sqrt{5}}{7}b_2 \end{pmatrix}$	$\frac{1}{2\sqrt{h_j}}\begin{pmatrix} 2a_0 & 2a_1 & 2a_2 \\ 2a_1 & 2a_0+\frac{4}{\sqrt{5}}a_2 & \frac{4}{\sqrt{5}}a_1 \\ 2a_2 & \frac{4}{\sqrt{5}}a_1 & 2a_0+\frac{4\sqrt{5}}{7}a_2 \end{pmatrix}$
3	$\frac{1}{2\sqrt{h_j}}\begin{pmatrix} 2b_0 & 2b_1 & 2b_2 & 2b_3 \\ 2b_1 & 2b_0+\frac{4}{\sqrt{5}}b_2 & \frac{4}{\sqrt{5}}b_1+6\sqrt{\frac{3}{35}}b_3 & 6\sqrt{\frac{3}{35}}b_2 \\ 2b_2 & \frac{4}{\sqrt{5}}b_1+6\sqrt{\frac{3}{35}}b_3 & 2b_0+\frac{4\sqrt{5}}{7}b_2 & 6\sqrt{\frac{3}{35}}b_1+\frac{8}{3\sqrt{5}}b_3 \\ 2b_3 & 6\sqrt{\frac{3}{35}}b_2 & 6\sqrt{\frac{3}{35}}b_1+\frac{8}{3\sqrt{5}}b_3 & 2b_0+\frac{8}{3\sqrt{5}}b_2 \end{pmatrix}$	$\frac{1}{2\sqrt{h_j}}\begin{pmatrix} 2a_0 & 2a_1 & 2a_2 & 2a_3 \\ 2a_1 & 2a_0+\frac{4}{\sqrt{5}}a_2 & \frac{4}{\sqrt{5}}a_1+6\sqrt{\frac{3}{35}}a_3 & 6\sqrt{\frac{3}{35}}a_2 \\ 2a_2 & \frac{4}{\sqrt{5}}a_1+6\sqrt{\frac{3}{35}}a_3 & 2a_0+\frac{4\sqrt{5}}{7}a_2 & 6\sqrt{\frac{3}{35}}a_1+\frac{8}{3\sqrt{5}}a_3 \\ 2a_3 & 6\sqrt{\frac{3}{35}}a_2 & 6\sqrt{\frac{3}{35}}a_1+\frac{8}{3\sqrt{5}}a_3 & 2a_0+\frac{8}{3\sqrt{5}}a_2 \end{pmatrix}$

Table A.2: **Computation of the block matrices J_1^j, J_2^j :** This table gives the expression of the block matrices J_1^j, J_2^j for all $k_j \in \{1,2,3\}$ and a given components of the species $\mathbf{Q}^+$, B on the interval I_j (respectively denoted a_l, b_l).

A.4.2 Update of the matrices N_1, J_1 and J_2

The goal in this section is to present how we efficiently update the matrices N_1, J_1 and J_2 at each step of the simulation of the EC' model. Let us introduce the following notations for a given $k \in \mathbb{N}$

- $\mathbb{I}_k$: the $(k+1) \times (k+1)$ identity matrix,
- $\mathbb{O}_k$: the $(k+1) \times (k+1)$ matrix with entries 0,
- $\mathcal{O}_k$: the $(k+1) \times 1$ matrix with entries 0.

Since the initial dimensional concentration $\tilde{C}^0_{\mathbf{Q}^+}$ of the specie $\mathbf{Q}^+$ is equal to zero, then according to (3.55), we have

$$a_l = 0, \forall l \in \{0, \cdots, k_j\}, \tag{A.9}$$

for all element $I_j, j \in \{1, \cdots, n\}$ at the time $\tilde{t} = 0$. Therefore from Tab A.1 and Tab A.2, the block matrix J_2^j and the block vector N_1^j take the form

$$J_2^j = \mathbb{O}_{k_j}, \quad N_1^j = \mathcal{O}_{k_j},\ j \in \{1, \cdots, n\}, \tag{A.10}$$

at the initial time $\tilde{t} = 0$. From (3.88), we have

$$b_l = \begin{cases} \frac{1}{2}\tilde{C}^0_{A^{total}}\sqrt{h_j} & \text{if } l = 0 \\ 0, & \text{if } l = 1, \cdots, k_j \end{cases}, \tag{A.11}$$

for all element $I_j, j \in \{1, \cdots, n\}$ at the time $\tilde{t} = 0$. Therefore from Tab A.2, the block matrix J_2^j takes the form

$$J_1^j = \frac{1}{2}\tilde{C}^0_{A^{total}}\mathbb{I}_{k_j},\ \ j \in \{1, \cdots, n\}. \tag{A.12}$$

Therefore at the initial time $\tilde{t} = 0$, we have

$$J_1 = \frac{1}{2}\tilde{C}^0_{A^{total}}\mathbb{I}_{n_T}, \quad J_2 = \mathbb{O}_{n_T}, \quad N_1 = \mathcal{O}_{n_T}, \tag{A.13}$$

for n_T given by (3.43).

Moreover at the boundary z_{max}, the species $\mathbf{Q}^+$ and B are subject to no flux condition. Then for every time $\tilde{t} = \tilde{t}_n$, there exist $j_n^{\mathbf{Q}^+}$ and j_n^B such that

- for all $j \geq j_n^{\mathbf{Q}^+}$, the component of $\mathbf{Q}^+$ on I_j is given by (A.9) and
- for all $j \geq j_n^B$, the component of B on I_j is given by (A.11).

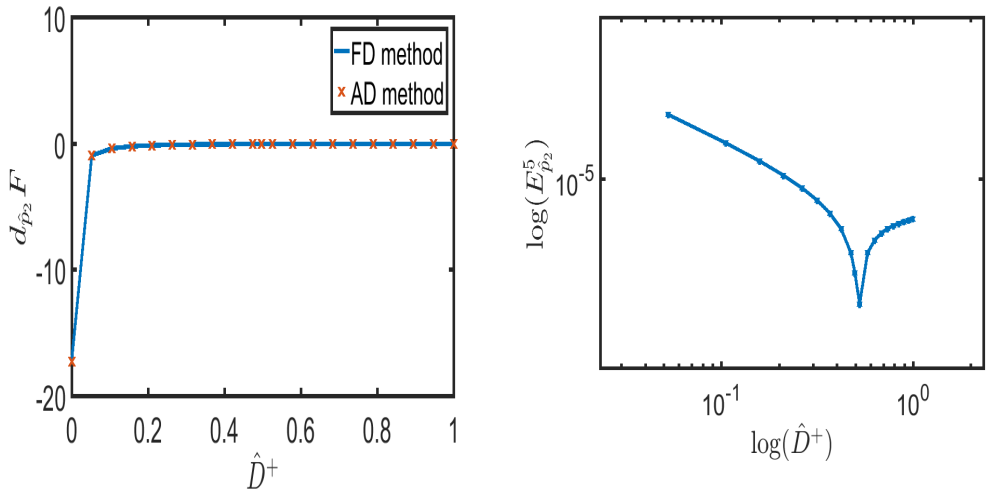
Therefore N_1^j, J_2^j are given by (A.10) for all $j \geq j_n^{\mathbf{Q}^+}$ and J_1^j is given by (A.12) for all $j \geq j_n^B$. Thus to compute N_1, J_2 and J_1 at the time $\tilde{t} = \tilde{t}_n$ from (A.13), we only need to update N_1^j for all $j < j_n^{\mathbf{Q}^+}$ with Tab A.1, J_2^j for all $j < j_n^{\mathbf{Q}^+}$ with Tab A.2 and J_1^j for all $j < j_n^B$ with Tab A.2.

A.5 Adjoint method for cyclic voltammetry models

In this section, we present the remaining tests for the comparison of the gradient of the objective function computed with the adjoint and finite difference method, while inverting numerically the cyclic voltammetry models.

A.5.1 Test of adjoint method for ETO model

The normalized model parameter of the ETO model is $\hat{p} = (\hat{K}_0, \hat{D}^+)^T$. Since the comparison of the total derivative $d_{\hat{K}_0} F$ obtained with the adjoint and finite difference methods has been investigated in Section 4.3.2, we focus here on the total derivative $d_{\hat{D}^+} F$. To this end, we use the same procedure described in Section 4.3.2 for the comparison of the total derivative $d_{\hat{K}_0} F$. The results is then illustrated in Fig A.1. We plot in Fig A.1(a) $d_{\hat{p}_2}^{n,r} F$ against the normalized parameter $\hat{p}_2^n$ for all $r = 1, 2$. We plot in Fig A.1(b) the difference $\log(E_{\hat{p}_2^n}^5)$ as a function of $\log(\hat{p}_2^n)$. We can conclude from Fig A.1(b) that the finite difference method and our implemented adjoint method leads approximatively to the same total derivative $d_{\hat{D}^+} F$.



(a) (b)

Figure A.1: **Comparaison of the gradient d_pF of ETO model using adjoint and finite difference method.** In (a) we plot the total derivative $d_{\hat{p}_2}F$, compute with the adjoint and finite difference method, as a function of the parameter $\hat{p}_2 = \hat{D}^+$. In (b) we plot the logarithm of the difference $E^5_{\hat{p}_2^n}$ as a function of the logarithm of $\hat{p}_2^n$.

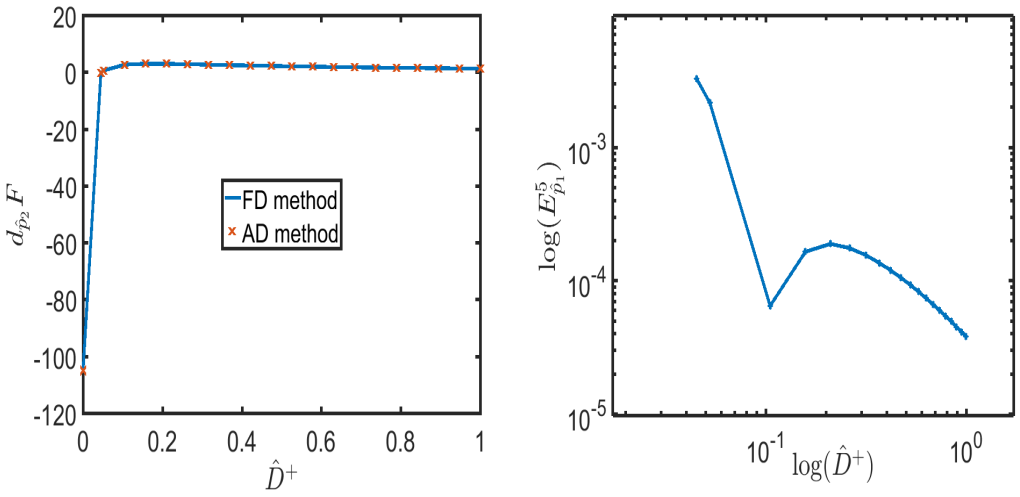
A.5.2 Test of adjoint method for EC' model

The normalized model parameter of the EC' model is given by

$$\hat{p} = (\hat{K}_0, \hat{D}^+, \hat{K}_r, \hat{D}^-, \hat{K}_A, \hat{D}, \hat{C}^0_{A^{total}}).$$

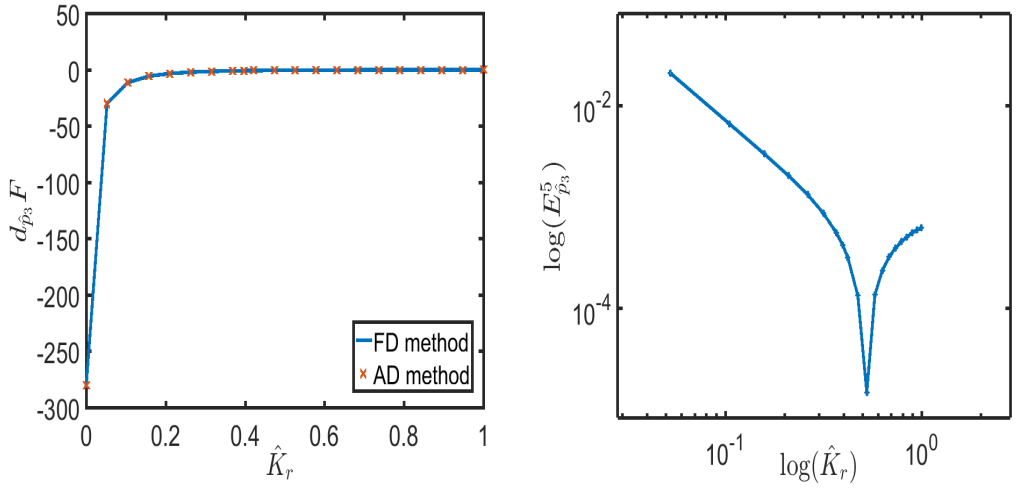
Since the comparison of the total derivative $d_{\hat{K}_0}F$ obtained with the adjoint and finite difference methods has been investigated in Section 4.4.2, we focus here on the total derivative of the objective function F with respect to entries $\hat{p}_i, i = 2, \cdots, 7$ of the model parameter $\hat{p}$ of the EC' model. To this end, we use the same procedure described in Section 4.4.2 to compare the total derivative $d_{\hat{p}_i}F$ obtained with the adjoint and finite difference method for all $i = 2, \cdots, 7$. The results is then illustrated in Fig A.2 to Fig A.7. We respectively plot in Fig A.2(a) to Fig A.7(a) $d^{n,r}_{\hat{p}_i}F$ against the normalized parameter $\hat{p}_i^n$ for all $r = 1, 2$ and $i = 2, \cdots, 7$. We respectively plot in Fig A.2(b) to Fig A.7(b) the difference $\log(E^5_{\hat{p}_i^n})$ as a function of $\log(\hat{p}_i^n)$ for all $i = 2, \cdots, 7$. We can conclude from Fig A.2(b) to Fig A.7(b) that the finite difference method and our implemented adjoint method leads approximatively

to the same total derivative $d_{\hat{p}_i} Fi = 2, \cdots, 7$.



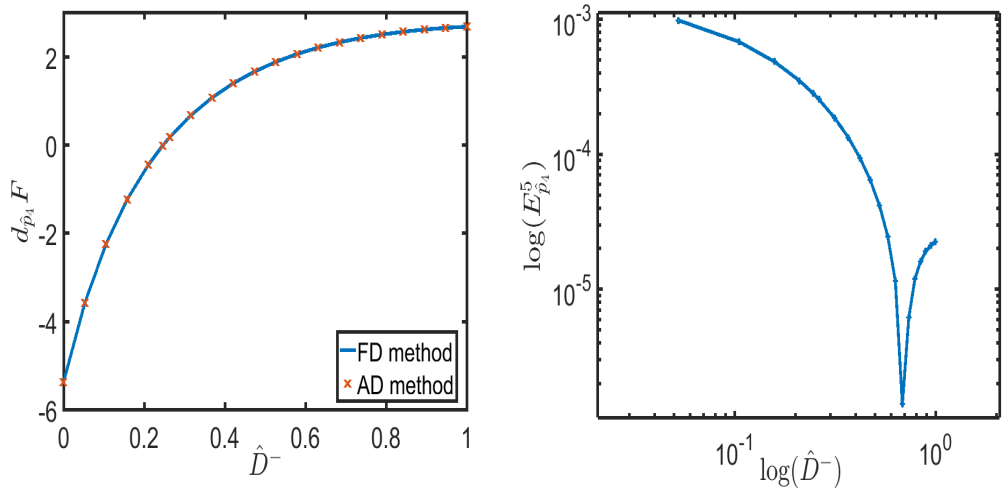
(a) (b)

Figure A.2: **Comparaison of the gradient $d_p F$ of EC' model using adjoint and finite difference method.** In (a) we plot the total derivative $d_{\hat{p}_2} F$, compute with the adjoint and finite difference method, as a function of the parameter $\hat{p}_2$. In (b) we plot the logarithm of the difference $E^5_{\hat{p}_2^n}$ as a function of the logarithm of $\hat{p}_2^n$.



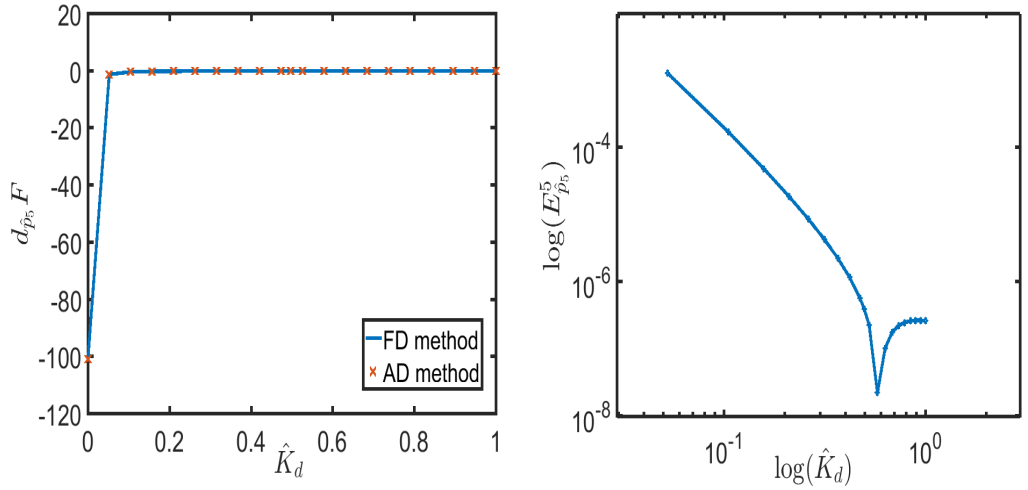
(a) (b)

Figure A.3: **Comparaison of the gradient $d_p F$ of EC' model using adjoint and finite difference method.** In (a) we plot the total derivative $d_{\hat{p}_3} F$, compute with the adjoint and finite difference method, as a function of the parameter $\hat{p}_3$. In (b) we plot the logarithm of the difference $E^5_{\hat{p}_3^n}$ as a function of the logarithm of $\hat{p}_3^n$.



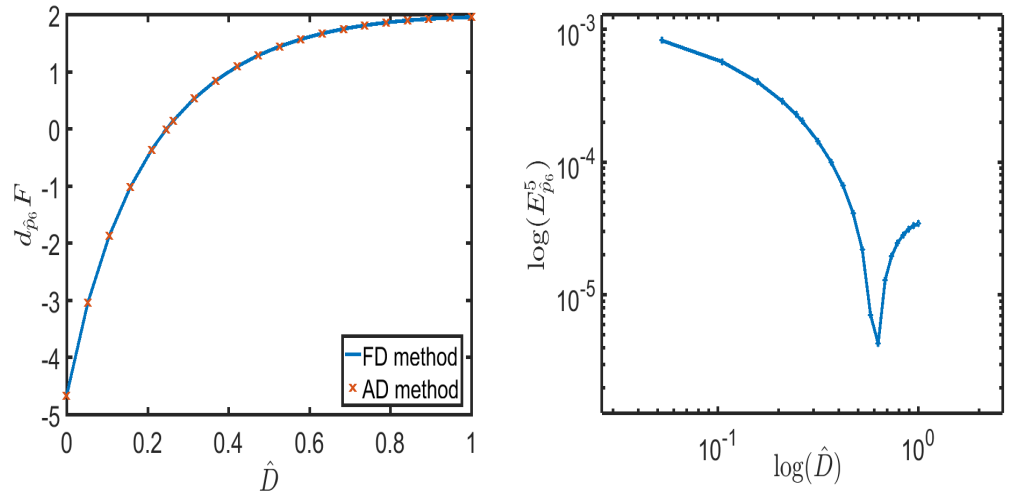
(a) (b)

Figure A.4: **Comparaison of the gradient d_pF of EC' model using adjoint and finite difference method.** In (a) we plot the total derivative $d_{\hat{p}_4}F$, compute with the adjoint and finite difference method, as a function of the parameter $\hat{p}_4$. In (b) we plot the logarithm of the difference $E^5_{\hat{p}_4^n}$ as a function of the logarithm of $\hat{p}_4^n$.



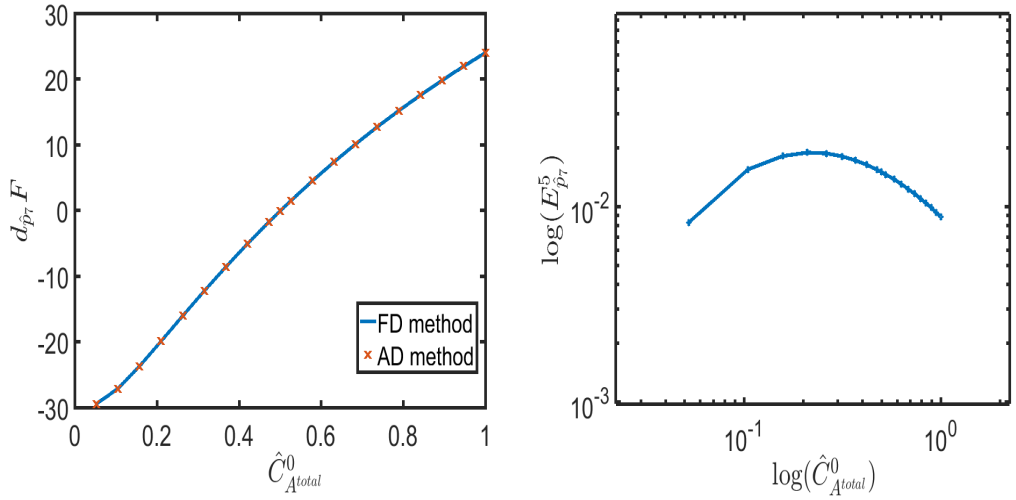
(a) (b)

Figure A.5: **Comparaison of the gradient d_pF of EC' model using adjoint and finite difference method.** In (a) we plot the total derivative $d_{\hat{p}_5}F$, compute with the adjoint and finite difference method, as a function of the parameter $\hat{p}_5$. In (b) we plot the logarithm of the difference $E^5_{\hat{p}_5^n}$ as a function of the logarithm of $\hat{p}_5^n$.



(a) (b)

Figure A.6: **Comparaison of the gradient d_pF of EC' model using adjoint and finite difference method.** In (a) we plot the total derivative $d_{\hat{p}_6}F$, compute with the adjoint and finite difference method, as a function of the parameter $\hat{p}_6$. In (b) we plot the logarithm of the difference $E^5_{\hat{p}^n_6}$ as a function of the logarithm of $\hat{p}^n_6$.



(a) (b)

Figure A.7: **Comparaison of the gradient d_pF of EC' model using adjoint and finite difference method.** In (a) we plot the total derivative $d_{\hat{p}_7}F$, compute with the adjoint and finite difference method, as a function of the parameter $\hat{p}_7$. In (b) we plot the logarithm of the difference $E^5_{\hat{p}^n_7}$ as a function of the logarithm of $\hat{p}^n_7$.

A.6 Implementation of the FE method

Here, we focus on how to efficiently assemble the mass and stiffness matrices derived from the FE discretization of the DAREs an unstructured triangulation. To that end, we first identify the non zeros entries of the mass and stiffness matrices. These entries, given by (2.9), are the sum of integral over the triangles of the triangulation. Finally, the non zero entries computed, by considering the non zeros integrals.

For a given $p, q \in \{1, \cdots, \mathcal{N}\}$, the entries $\mathbf{M}(p,q)$ and $\mathbf{S}(p,q)$ are non zero if and only if the intersection of the set S_p and S_q contains at least one element $T \in \mathcal{T}$. In the case $p = q$, the diagonal entries of $\mathbf{M}$ and $\mathbf{S}$ are obviously non zero and can be computed as

$$\mathbf{M}(p,p) = \sum_{T \in S_p} m^T(\phi_p, \phi_p), \quad \mathbf{S}(p,p) = \sum_{T \in S_p} a^T(\phi_p, \phi_q),$$

for all $p \in \{1, \cdots, \mathcal{N}\}$. In the case $p \neq q$, the intersection of S_p and S_q is not empty if only there is an edge $F \in \mathcal{F}$ such that $F = F^{pq}$ where F^{pq} represent the segment linking the nodes N_p and N_q. Thus non diagonal entries of $\mathbf{M}$ and $\mathbf{S}$ that are non zero can be computed throughout the edges $F \in \mathcal{F}$. If the edge F belongs to $\mathcal{F}_i$, then there exists two element $T_j, T_n \in \mathcal{T}$ such that $F = \partial T_j \cap \partial T_n$; and we have

$$\mathbf{M}(p,q) = \mathbf{M}(q,p) = \sum_{i=j,n} m^{T_i}(\phi_q, \phi_p), \ \mathbf{S}(p,q) = \sum_{i=j,n} a^{T_i}(\phi_q, \phi_p), \ \mathbf{S}(q,p) = \sum_{i=j,n} a^{T_i}(\phi_p, \phi_q).$$

If the edge F belongs to $\mathcal{F}_e$, then there exists only one element $T_j \in \mathcal{T}$ such that $F = \partial T_j \cap \partial \Omega$; and we have

$$\mathbf{M}(p,q) = \mathbf{M}(q,p) = m^{T_j}(\phi_q, \phi_p), \quad \mathbf{S}(p,q) = a^{T_j}(\phi_q, \phi_p), \quad \mathbf{S}(q,p) = a^{T_j}(\phi_p, \phi_q).$$

This shows that neither the mass nor stiffness matrices is diagonal or block diagonal. But since the bilinear form $m^T(\cdot,\cdot)$ for a given element T is symmetric, the mass matrix is symmetric. Therefore the mass and stiffness matrices can be efficiently assembled with Algorithm 7.

Algorithm 7: Efficient method to assemble $\mathbf{M}$ and $\mathbf{S}$ from FE method.

Initialize $\mathbf{M}$ and $\mathbf{S}$ as sparse matrices
for $p = 1, \cdots, \mathcal{N}$ do (Loop over node N_p)
  find support S_p of ϕ_p
  for $i = 1, \cdots, r_p$ do
    $\mathbf{M}(p,p) \longrightarrow \mathbf{M}(p,p) + m^{T_i}(\phi_p, \phi_p)$
    $\mathbf{S}(p,p) \longrightarrow \mathbf{S}(p,p) + a^{T_i}(\phi_p, \phi_p)$
  end for
end for
for $F^{pq} \in \mathcal{F}^i$ i.e. $F^{pq} = \partial T_j \cap \partial T_n$ do (Loop over internal edges)
  for $i = j, n$ do
    $\mathbf{S}(p,q) \longrightarrow \mathbf{S}(p,q) + a^{T_i}(\phi_q, \phi_p)$
    $\mathbf{S}(q,p) \longrightarrow \mathbf{S}(q,p) + a^{T_i}(\phi_p, \phi_q)$
    $\mathbf{M}(p,q) \longrightarrow \mathbf{M}(p,q) + m^{T_i}(\phi_q, \phi_p)$
  end for
  $\mathbf{M}(q,p) \longrightarrow \mathbf{M}(p,q)$
end for
for $F^{pq} \in \mathcal{F}^e$ i.e. $F^{pq} = \partial T_j \cap \partial\Omega$ do (Loop over external edges)
  $\mathbf{S}(p,q) \longrightarrow \mathbf{S}(p,q) + a^{T_j}(\phi_q, \phi_p)$
  $\mathbf{S}(q,p) \longrightarrow \mathbf{S}(q,p) + a^{T_j}(\phi_p, \phi_q)$
  $\mathbf{M}(p,q) \longrightarrow \mathbf{M}(p,q) + m^{T_j}(\phi_q, \phi_p)$
  $\mathbf{M}(q,p) \longrightarrow \mathbf{M}(p,q)$
end for

Once the matrices $\mathbf{M}, \mathbf{S}$ are assembled, (2.8) can been solved with several time step solver.

Bibliography

[1] A. Abdulle and G. Vilmart. PIROCK: a swiss-knife partitioned implicit-explicit orthogonal Runge-Kutta Chebyshev integrator for stiff diffusion-advection-reaction problems with or without noise. *J. Comput. Phys.*, 242:869–888, 2013.

[2] J. Adams. on the expression for product of any two legendreś coefficients by means of series of legendreś coefficient. *Proc. Roy. Soc. LONDON*, 27:63–71, 1878.

[3] S. S. S. Ahmadi, B. Raissi, R. Riahifar, M. S. Yaghmaee, H. R. Saadati, S. Ghashghaie, and M. Javaheri. Fabrication of counter electrode of electrochemical co gas sensor by electrophoretic deposition of mwcnt. *Journal of the Electrochemical Society*, 162:D3101, 2015.

[4] G. Akrivis and C. Makridakis. Galerkin time-stepping methods for nonlinear parabolic equations. *ESAIM: Mathematical Modelling and Numerical Analysis*, 38(2):261–289, 2004.

[5] A. H. Al-Mohy and N. J. Higham. A new scaling and squaring algorithm for the matrix exponential. *SIAM J. Matrix Anal. Appl.*, 31(3):970–989, 2009.

[6] W. A. Al-Salam. On the product of two Legendre polynomials. *Math. Scand.*, 4:239–242, 1956.

[7] G. Alessandrini and G. Uhlmann, editors. *Inverse problems: theory and applications*, volume 333 of *Contemporary Mathematics*. American Mathematical Society, Providence, RI, 2003.

[8] L. Angulo, J. Alvarez, F. L. Teixeira, M. F. Pantoja, and S. G. Garcia. Causal-path local time-stepping in the discontinuous Galerkin method for Maxwells equations. *Journal of Computational Physics*, 256:678–695, 2014.

[9] K. Aoki, K. Tokuda, and H. Matsuda. Theory of linear sweep voltammetry with finite diffusion space part ii. totally irreversible and quasi-reversible cases. *Journal of Electroanalytical Chemistry*, 160(1-2):3345, Oct 1984.

[10] D. N. Arnold. An interior penalty finite element method with discontinuous elements. *SIAM journal on numerical analysis*, 19(4):742–760, 1982.

[11] D. N. Arnold. An interior penalty finite element method with discontinuous elements. *SIAM J. Numer. Anal.*, 19(4):742–760, 1982.

[12] H. Ashi. *Numerical Methods for Stiff Systems*. Phd, University of Nottingham, 2008.

[13] B. Ayuso and L. D. Marini. Discontinuous Galerkin methods for advection-diffusion-reaction problems. *SIAM Journal on Numerical Analysis*, 47(2):1391–1420, 2009.

[14] I. Babuška. The finite element method with penalty. *Mathematics of computation*, 27(122):221–228, 1973.

[15] I. Babuška and M. Zlámal. Nonconforming elements in the finite element method with penalty. *SIAM Journal on Numerical Analysis*, 10(5):863–875, 1973.

[16] J. Baglama, D. Calvetti, and L. Reichel. Fast Leja points. *Electron. Trans. Numer. Anal.*, 7:124–140, 1998. Large scale eigenvalue problems (Argonne, IL, 1997).

[17] G. A. Baker. Finite element methods for elliptic equations using nonconforming elements. *Mathematics of Computation*, 31(137):45–59, 1977.

[18] M. Balázsová and M. Feistauer. On the stability of the ale space-time discontinuous Galerkin method for nonlinear convection-diffusion problems in time-dependent domains. *Applications of Mathematics*, 60(5):501–526, 2015.

[19] L. Banjai, E. H. Georgoulis, and O. Lijoka. A trefftz polynomial space-time discontinuous Galerkin method for the second order wave equation. *SIAM Journal on Numerical Analysis*, 55(1):63–86, 2017.

[20] O. Banton and L. Bangoy. Hydrogeologie, multiscience environnementale des eaux souterraines. *Presse de l'Universite de Québec*, 1997.

[21] J. A. Barceló, L. Vega, and M. Zubeldia. The forward problem for the electromagnetic Helmholtz equation with critical singularities. *Adv. Math.*, 240:636–671, 2013.

[22] A. Bard and L. Faulkner. *Electrochemical Methods: Fundamentals and Applications*. Wiley, 2000.

[23] F. Bashforth and J. C. Adams. *An attempt to test the theories of capillary action: by comparing the theoretical and measured forms of drops of fluid. With an explanation of the method of integration employed in constucting the tables which give the theoretical forms of such drops*. University Press, 1883.

[24] F. Bassi and S. Rebay. A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier-Stokes equations. *J. Comput. Phys.*, 131(2):267–279, 1997.

[25] K.-J. Bathe. *Finite element method*. Wiley Online Library, 2008.

[26] J. Bear. *Dynamics of Fluids in Porous Media*. Dover Civil and Mechanical Engineering Series. Dover, 1972.

[27] P. Bedient, H. Rifai, and C. Newell. *Ground Water Contamination: Transport and Remediation*. Prentice Hall PTR, 1999.

[28] G. Beni. Short-circuit memory in electrochromic displays. *Appl. Phys. Lett. Applied Physics Letters*, 37(1):106, 1980.

[29] L. Bergamaschi, M. Caliari, and M. Vianello. The ReLPM exponential integrator for FE discretizations of advection-diffusion equations. In *Computational science—ICCS 2004. Part IV*, volume 3039 of *Lecture Notes in Comput. Sci.*, pages 434–442. Springer, Berlin, 2004.

[30] H. Berland and B. Skaflestad. Solving the nonlinear schrodinger equation using exponential integrators. *Norwegian Society of Automatic Control*, 27:201–217, 2006.

[31] H. Blum, S. Lisky, and R. Rannacher. A domain splitting algorithm for parabolic problems. *Computing*, 49(1):11–23, 1992.

[32] J. F. Bonnans, J. C. Gilbert, C. Lemaréchal, and C. A. Sagastizábal. *Numerical optimization*. Universitext. Springer-Verlag, Berlin, second edition, 2006. Theoretical and practical aspects.

[33] A. Bott. Simulation of cyclic voltammetry using finite difference methods. *Current Separations*, 19(2):45–48, 2000.

[34] A. W. Bott. Characterization of chemical reactions coupled to electron transfer reactions using cyclic voltammetry. *Current Separations*, 18(1):9–16, 1999.

[35] A. W. Bott. Investigation of enzyme-mediated electron transfer using digisim®. *Current Separations*, 20(4), 2004.

[36] A. W. Bott, S. W. Feldberg, and M. Rudolph. Fitting experimental cyclic voltammetry data with theoretical simulations using digisim [r] 2.1. *Current Separations*, 15:67–71, 1996.

[37] K. Böttcher and R. Rannacher. *Adaptive error control in solving ordinary differential equations by the discontinuous Galerkin method*. Universität Heidelberg. Interdisziplinäres Zentrum für Wissenschaftliches Rechnen [IWR], 1996.

[38] S. C. Brenner and L. R. Scott. *The mathematical theory of finite element methods*, volume 15 of *Texts in Applied Mathematics*. Springer, New York, third edition, 2008.

[39] D. Britz. *Digital Simulation in Electrochemistry*. Springer, 2010.

[40] M. Brun, A. Lallemand, J.-F. Quinson, and C. Eyraud. A new method for the simultaneous determination of the size and shape of pores: the thermoporometry. *Thermochimica Acta*, 21(1):59 – 88, 1977.

[41] R. L. Burden, J. D. Faires, and A. C. Reynolds. *Numerical analysis*. Prindle, Weber & Schmidt, Boston, Mass., 1978.

[42] E. Burman, A. Ern, I. Mozolevski, and B. Stamm. The symmetric discontinuous Galerkin method does not need stabilization in 1D for polynomial orders $p \geq 2$. *C. R. Math. Acad. Sci. Paris*, 345(10):599–602, 2007.

[43] E. Burman and P. Zunino. A domain decomposition method based on weighted interior penalties for advection-diffusion-reaction problems. *SIAM Journal on Numerical Analysis*, 44(4):1612–1638, 2006.

[44] J. C. Butcher. *Numerical methods for ordinary differential equations*. John Wiley & Sons, 2016.

[45] R. H. Byrd, J. C. Gilbert, and J. Nocedal. A trust region method based on interior point techniques for nonlinear programming. *Math. Program.*, 89(1, Ser. A):149–185, 2000.

[46] R. H. Byrd, M. E. Hribar, and J. Nocedal. An interior point algorithm for large-scale nonlinear programming. *SIAM J. Optim.*, 9(4):877–900, 1999. Dedicated to John E. Dennis, Jr., on his 60th birthday.

[47] M. Caliari, M. Vianello, and L. Bergamaschi. Interpolating discrete advection-diffusion propagators at Leja sequences. *J. Comput. Appl. Math.*, 172(1):79–99, 2004.

[48] M. P. Calvo and C. Palencia. A class of explicit multistep exponential integrators for semilinear problems. *Numer. Math.*, 102(3):367–381, 2006.

[49] Y. Cao, S. Li, L. Petzold, and R. Serban. Adjoint sensitivity analysis for differential-algebraic equations: the adjoint DAE system and its numerical solution. *SIAM J. Sci. Comput.*, 24(3):1076–1089 (electronic), 2002.

[50] J. Česenek and M. Feistauer. Theory of the space-time discontinuous Galerkin method for nonstationary parabolic problems with nonlinear convection and diffusion. *SIAM Journal on Numerical Analysis*, 50(3):1181–1206, 2012.

[51] G. Chavent, B. Cockburn, G. Cohen, and J. Jaffré. Une méthode d'éléments finis pour la simulation dans un réservoir de déplacements bidimensionnel d'huile par de l'eau. *INRIA*, 353, 1985.

[52] G. Chavent, G. Cohen, and J. Jaffré. A finite element simulator for incompressible two-phase flow. *Transport in Porous Media*, 2(5):465–478, 1987.

[53] N. Chevaugeon, J. Xin, P. Hu, X. Li, D. Cler, J. E. Flaherty, and M. S. Shephard. Discontinuous Galerkin methods applied to shock and blast problems. *J. Sci. Comput.*, 22/23:227–243, 2005.

[54] K. Chrysafinos and N. J. Walkington. Error estimates for discontinuous Galerkin approximations of implicit parabolic equations. *SIAM journal on numerical analysis*, 43(6):2478–2499, 2006.

[55] K. Chrysafinos and N. J. Walkington. Error estimates for the discontinuous Galerkin methods for parabolic equations. *SIAM Journal on Numerical Analysis*, 44(1):349–366, 2006.

[56] P. G. Ciarlet. *The finite element method for elliptic problems*. SIAM, 2002.

[57] B. Cockburn, B. Dong, J. Guzmán, M. Restelli, and R. Sacco. A hybridizable discontinuous Galerkin method for steady-state convection-diffusion-reaction problems. *SIAM Journal on Scientific Computing*, 31(5):3827–3846, 2009.

[58] B. Cockburn, G. Kanschat, and D. Schötzau. The local discontinuous Galerkin method for linearized incompressible fluid flow: a review. *Computers & fluids*, 34(4):491–506, 2005.

[59] B. Cockburn, G. E. Karniadakis, and C.-W. Shu. The development of discontinuous Galerkin methods. In *Discontinuous Galerkin Methods*, pages 3–50. Springer, 2000.

[60] B. Cockburn, G. E. Karniadakis, and C. W. Shu. *Discontinuous Galerkin Methods: Theory, Computation and Applications*, volume 11 of *Lecture notes in computational science and engineering*. Springer Publishing Company, Incorporated, 2000.

[61] B. Cockburn, S.-Y. Lin, and C.-W. Shu. Tvb runge-kutta local projection discontinuous Galerkin finite element method for conservation laws III: one-dimensional systems. *Journal of Computational Physics*, 84(1):90–113, 1989.

[62] B. Cockburn and C.-W. Shu. TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws. II. General framework. *Math. Comp.*, 52(186):411–435, 1989.

[63] B. Cockburn and C.-W. Shu. The local discontinuous Galerkin method for time-dependent convection-diffusion systems. *SIAM J. Numer. Anal.*, 35(6):2440–2463 (electronic), 1998.

[64] B. Cockburn and C.-W. Shu. The runge-kutta discontinuous Galerkin method for conservation laws V: multidimensional systems. *Journal of Computational Physics*, 141(2):199–224, 1998.

[65] R. Codina. Comparison of some finite element methods for solving the diffusion-convection-reaction equation. *Computer Methods in Applied Mechanics and Engineering*, 156(1-4):185–210, 1998.

[66] R. G. Compton, E. Laborda, and K. R. Ward. *Understanding voltammetry: simulation of electrode processes*, volume 3 of *Understanding voltammetry*. Imperial College Press, 2014.

[67] R. Courant, K. Friedrichs, and H. Lewy. On the partial difference equations of mathematical physics. *IBM J. Res. Develop.*, 11:215–234, 1967.

[68] S. M. Cox and P. C. Matthews. Exponential time differencing for stiff systems. *J. Comput. Phys.*, 176(2):430–455, 2002.

[69] J. Crank. *The mathematics of diffusion.* Clarendon Press, Oxford, second edition, 1975.

[70] G. Dahlquist. Convergence and stability in the numerical integration of ordinary differential equations. *Mathematica Scandinavica*, pages 33–53, 1956.

[71] G. G. Dahlquist. A special stability problem for linear multistep methods. *BIT Numerical Mathematics*, 3(1):27–43, 1963.

[72] Darcy's experiment. `https://www.flickr.com/photos/mitopencourseware/sets/72157614684644687/`.

[73] C. N. Dawson and Q. Du. A finite element domain decomposition method for parabolic equations. In *Fourth International Symposium on Domain Decomposition Methods for PDEs, edited by R. Glowinski et al, pp255-263, SIAM, Philadelphia*, 1991.

[74] C. N. Dawson, Q. Du, and T. F. Dupont. A finite difference domain decomposition algorithm for numerical solution of the heat equation. *Math. Comp.*, 57(195):63–71, 1991.

[75] C. N. Dawson and T. F. Dupont. Explicit/implicit conservative Galerkin domain decomposition procedures for parabolic problems. *mathematics of computation*, 58(197):21–34, 1992.

[76] M. Delfour, W. Hager, and F. Trochu. Discontinuous Galerkin methods for ordinary differential equations. *Mathematics of Computation*, 36(154):455–473, 1981.

[77] J. V. der Vegt and H. V. der Ven. Space-time discontinuous Galerkin finite element method with dynamic grid motion for inviscid compressible flows: I. general formulation. *Journal of Computational Physics*, 182(2):546–585, 2002.

[78] J. V. der Vegt and H. V. der Ven. Space-time discontinuous Galerkin finite element method with dynamic grid motion for inviscid compressible flows: Ii. efficient flux quadrature. *Computer methods in applied mechanics and engineering*, 191(41):4747–4780, 2002.

[79] G. Dhatt, E. LefranÃ, G. Touzot, et al. *Finite element method*. John Wiley & Sons, 2012.

[80] D. A. Di Pietro and A. Ern. *Mathematical aspects of discontinuous Galerkin methods*, volume 69 of *Mathematiques and Applications (Berlin)*. Springer, Heidelberg, 2012.

[81] D. A. Di Pietro, A. Ern, and J.-L. Guermond. Discontinuous Galerkin methods for anisotropic semidefinite diffusion with advection. *SIAM Journal on Numerical Analysis*, 46(2):805–831, 2008.

[82] J. Diaz and M. J. Grote. Energy conserving explicit local time stepping for second-order wave equations. *SIAM Journal on Scientific Computing*, 31(3):1985–2014, 2009.

[83] J. Diaz and M. J. Grote. Multi-level explicit local time-stepping methods for second-order wave equations. *Computer methods in applied mechanics and engineering*, 291:240–265, 2015.

[84] J. Dolbow, T. Belytschko, et al. A finite element method for crack growth without remeshing. *International journal for numerical methods in engineering*, 46(1):131–150, 1999.

[85] V. Dolejší and M. Feistauer. Discontinuous Galerkin method. *Analysis and Applications to Compressible Flow*, 48, 2015.

[86] J. Douglas, Jr. and T. Dupont. Interior penalty procedures for elliptic and parabolic Galerkin methods. pages 207–216. Lecture Notes in Phys., Vol. 58, 1976.

[87] M. Dryja. On discontinuous Galerkin methods for elliptic problems with discontinuous coefficients. *Computational Methods in Applied Mathematics Comput. Methods Appl. Math.*, 3(1):76–85, 2003.

[88] M. Dryja et al. Substructuring methods for parabolic problems. In *Proceedings of the Fourth International Symposium on Domain Decomposition Methods for Partial Differential Equations*, pages 264–271, 1991.

[89] F. DULLIEN. 1 - pore structure. In F. DULLIEN, editor, *Porous Media (Second Edition)*, pages 5 – 115. Academic Press, San Diego, second edition edition, 1992.

[90] M. Dumbser, M. Käser, and E. F. Toro. An arbitrary high-order discontinuous Galerkin method for elastic waves on unstructured meshes-v. local time stepping and p-adaptivity. *Geophysical Journal International*, 171(2):695–717, 2007.

[91] M. Dumbser, M. Kser, and E. F. Toro. An arbitrary high-order discontinuous Galerkin method for elastic waves on unstructured meshes v. local time stepping and p-adaptivity. *Geophysical Journal International*, 171(2):695–717, 2007.

[92] R. G. Duran. Galerkin approximations and finite element methods. *Lecture notes at ICTP*, 1996.

[93] H. W. Engl, K. Kunisch, and A. Neubauer. Convergence rates for Tikhonov regularisation of nonlinear ill-posed problems. *Inverse Problems*, 5(4):523–540, 1989.

[94] C. Engstler and C. Lubich. Multirate extrapolation methods for differential equations with different time scales. *Computing*, 58(2):173–185, 1997.

[95] K. Eriksson and C. Johnson. Adaptive finite element methods for parabolic problems i: A linear model problem. *SIAM Journal on Numerical Analysis*, 28(1):43–77, 1991.

[96] K. Eriksson, C. Johnson, and A. Logg. *Adaptive computational methods for parabolic problems*. Wiley Online Library, 2004.

[97] A. Ern and J.-L. Guermond. *Theory and practice of finite elements*, volume 159 of *Applied Mathematical Sciences*. Springer-Verlag, New York, 2004.

[98] A. Ern, A. F. Stephansen, and P. Zunino. A discontinuous Galerkin method with weighted averages for advection–diffusion equations with locally small and anisotropic diffusivity. *IMA Journal of Numerical Analysis*, 29(2):235–256, 2008.

[99] D. Estep. A posteriori error bounds and global error control for approximation of ordinary differential equations. *SIAM Journal on Numerical Analysis*, 32(1):1–48, 1995.

[100] D. Estep and S. Larsson. The discontinuous Galerkin method for semilinear parabolic problems. *RAIRO-Modélisation mathématique et analyse numérique*, 27(1):35–54, 1993.

[101] L. C. Evans. *Partial differential equations*, volume 19 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, 1998.

[102] R. E. Ewing, R. D. Lazarov, and A. T. Vassilev. Finite difference scheme for parabolic problems on composite grids with refinement in time and space. *SIAM Journal on Numerical Analysis*, 31(6):1605–1622, 1994.

[103] I. Faille, F. Nataf, F. Willien, and S. Wolf. Two local time stepping schemes for parabolic problems. In *Multiresolution and adaptive methods for convection-dominated problems*, volume 29 of *ESAIM Proc.*, pages 58–72. EDP Sci., Les Ulis, 2009.

[104] I. Faille, F. Nataf, F. Willien, and S. Wolf. Two local time stepping schemes for parabolic problems. In *ESAIM: proceedings*, volume 29, pages 58–72. EDP Sciences, 2009.

[105] F. Fambri, M. Dumbser, and O. Zanotti. Space-time adaptive ader-dg schemes for dissipative flows: Compressible Navier-Stokes and resistive mhd equations. *Computer Physics Communications*, 220:297–318, 2017.

[106] M. Feistauer, J. Hájek, and K. Švadlenka. Space-time discontinuos Galerkin method for solving nonstationary convection-diffusion-reaction problems. *Applications of Mathematics*, 52(3):197–233, 2007.

[107] M. Feistauer, V. Kučera, K. Najzar, and J. Prokopová. Analysis of space–time discontinuous Galerkin method for nonlinear convection–diffusion problems. *Numerische Mathematik*, 117(2):251–288, 2011.

[108] M. Feistauer, V. Kučera, K. Najzar, and J. Prokopová. Analysis of space-time discontinuous Galerkin method for nonlinear convection-diffusion problems. *Numerische Mathematik*, 117(2):251–288, 2011.

[109] A. Fick. Ueber diffusion. *Annalen der Physik*, 170(1):59–86, 1855.

[110] A. Fick. On liquid diffusion. *Journal of Membrane Science*, 100(1):33–38, 1995.

[111] L. P. Franca and F. Valentin. On an improved unusual stabilized finite element method for the advective–reactive–diffusive equation. *Computer Methods in Applied Mechanics and Engineering*, 190(13):1785–1800, 2000.

[112] W. Freeden and H. Nutz. Satellite gravity gradiometry as tensorial inverse problem. *GEM Int. J. Geomath.*, 2(2):177–218, 2011.

[113] A. Galeao, R. Almeida, S. Malta, and A. Loula. Finite element analysis of convection dominated reaction–diffusion problems. *Applied Numerical Mathematics*, 48(2):205–222, 2004.

[114] M. J. Gander and C. Rohde. Overlapping Schwarz waveform relaxation for convection-dominated nonlinear conservation laws. *SIAM journal on Scientific Computing*, 27(2):415–439, 2005.

[115] C. W. Gear and D. Wells. Multirate linear multistep methods. *BIT Numerical Mathematics*, 24(4):484–502, 1984.

[116] E. H. Georgoulis, E. Hall, and P. Houston. Discontinuous Galerkin methods for advection-diffusion-reaction problems on anisotropically refined meshes. *SIAM Journal on Scientific Computing*, 30(1):246–271, 2007.

[117] F. X. Giraldo. High-order triangle-based discontinuous Galerkin methods for hyperbolic equations on a rotating sphere. *J. Comput. Phys.*, 214(2):447–465, 2006.

[118] F. X. Giraldo, J. S. Hesthaven, and T. Warburton. Nodal high-order discontinuous Galerkin methods for the spherical shallow water equations. *J. Comput. Phys.*, 181(2):499–525, 2002.

[119] G. M. L. Gladwell and A. Morassi, editors. *Dynamical inverse problems: theory and application*, volume 529 of *CISM Courses and Lectures*. SpringerWienNewYork, Vienna, 2011. Papers from the CISM Course held in Udine, May 25–29, 2009.

[120] G. H. Golub and C. F. Van Loan. *Matrix computations*, volume 3 of *Johns Hopkins Series in the Mathematical Sciences*. Johns Hopkins University Press, Baltimore, MD, 1983.

[121] C. Günther and P. Henkel. Integer ambiguity estimation for satellite navigation. *IEEE Trans. Signal Process.*, 60(7):3387–3393, 2012.

[122] P. C. Hansen. Perturbation bounds for discrete Tikhonov regularisation. *Inverse Problems*, 5(4):L41–L44, 1989.

[123] G. H. Hardy and E. M. Wright. *An introduction to the theory of numbers*. The Clarendon Press, Oxford University Press, New York, fifth edition, 1979.

[124] D. Henry. *Geometric theory of semilinear parabolic equations*, volume 840 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin-New York, 1981.

[125] J. S. Hesthaven and T. Warburton. *Nodal discontinuous Galerkin methods*, volume 54 of *Texts in Applied Mathematics*. Springer, New York, 2008. Algorithms, analysis, and applications.

[126] N. J. Higham. The scaling and squaring method for the matrix exponential revisited. *SIAM J. Matrix Anal. Appl.*, 26(4):1179–1193 (electronic), 2005.

[127] K. Hillewaert, N. Chevaugeon, P. Geuzaine, and J.-F. Remacle. Hierarchic multigrid iteration strategy for the discontinuous Galerkin solution of the steady Euler equations. *Internat. J. Numer. Methods Fluids*, 51(9-10):1157–1176, 2006.

[128] M. Hochbruck and C. Lubich. On Krylov subspace approximations to the matrix exponential operator. *SIAM J. Numer. Anal.*, 34(5):1911–1925, 1997.

[129] M. Hochbruck, C. Lubich, and H. Selhofer. Exponential integrators for large systems of differential equations. *SIAM J. Sci. Comput.*, 19(5):1552–1574 (electronic), 1998.

[130] M. Hochbruck and A. Ostermann. Exponential Runge-Kutta methods for parabolic problems. *Appl. Numer. Math.*, 53(2-4):323–339, 2005.

[131] M. Hochbruck and A. Ostermann. Exponential integrators. *Acta Numer.*, 19:209–286, 2010.

[132] M. Hochbruck, A. Ostermann, and J. Schweitzer. Exponential Rosenbrock-type methods. *SIAM J. Numer. Anal.*, 47(1):786–803, 2008/09.

[133] P. Houston, C. Schwab, and E. Süli. Discontinuous hp-finite element methods for advection-diffusion-reaction problems. *SIAM Journal on Numerical Analysis*, 39(6):2133–2163, 2002.

[134] M. HUBBERT. Darcy's law and the field equations of the flow of underground fluids. *International Association of Scientific Hydrology. Bulletin*, 2(1):23–59, 1957.

[135] T. J. Hughes. *The finite element method: linear static and dynamic finite element analysis*. Courier Corporation, 2012.

[136] B. L. Hulme. One-step piecewise polynomial Galerkin methods for initial value problems. *Mathematics of Computation*, 26(118):415–426, 1972.

[137] B. L. Hulme. One-step piecewise polynomial Galerkin methods for initial value problems. *Mathematics of Computation*, 26(118):881–891, 1972.

[138] T. Inoue, A. Fujishima, and K. Honda. Photoelectrochromic characteristics of photoelectrochemical imaging system with a semiconductor/solution (metallic ion) junction. *Journal of The Electrochemical Society J. Electrochem. Soc.*, 127(7):1582, 1980.

[139] V. John and E. Schmeyer. Finite element methods for time-dependent convection–diffusion–reaction equations with small diffusion. *Computer methods in applied mechanics and engineering*, 198(3):475–494, 2008.

[140] A. Johnson, G. Merilis, J. Hastings, M. E. Palmer, J. P. Fitts, and D. Chidambaram. Reductive degradation of organic compounds using microbial nanotechnology. *Journal of the Electrochemical Society*, 160(1), Jul 2012.

[141] C. Johnson. Error estimates and adaptive time-step control for a class of one-step methods for stiff ordinary differential equations. *SIAM Journal on Numerical Analysis*, 25(4):908–926, 1988.

[142] C. Johnson, U. Nävert, and J. Pitkäranta. Finite element methods for linear hyperbolic problems. *Computer methods in applied mechanics and engineering*, 45(1-3):285–312, 1984.

[143] J. Kenney and E. Keeping. *Mathematics of statistics*. Number pt. 2 in Mathematics of Statistics. Van Nostrand, 1947.

[144] S. Kim, R. L. Wang, A. K. Khambampati, B. A. Lee, and K. Y. Kim. An improved boundary distributed source method for electrical resistance tomography forward problem. *Eng. Anal. Bound. Elem.*, 44:185–192, 2014.

[145] C. M. Klaij, J. J. van der Vegt, and H. van der Ven. Space–time discontinuous Galerkin method for the compressible Navier-Stokes equations. *Journal of Computational Physics*, 217(2):589–611, 2006.

[146] M. J. Klein, K. Goossens, C. W. Bielawski, and A. Manthiram. Elucidating the electrochemical activity of electrolyte-insoluble polysulfide species in lithium-sulfur batteries. *Journal of The Electrochemical Society J. Electrochem. Soc.*, 163(9), 2016.

[147] D. Kuzmin. *A Guide to Numerical Methods For Transport Equations*. PhD thesis, Department of maths, TU Dortmund University, Mar. 2010.

[148] Y. M. Laevsky. On the domain decomposition method for parabolic problems. *Bull. Novosibirsk Comput. Center*, 1:41–62, 1993.

[149] J. D. Lawson. Generalized Runge-Kutta processes for stable systems with large Lipschitz constants. *SIAM J. Numer. Anal.*, 4:372–380, 1967.

[150] V. I. Lebedev. Explicit difference schemes with time-variable steps for solving stiff systems of equations. *Soviet J. Numer. Anal. Math. Modelling*, 4(2):111–135, 1989.

[151] V. I. Lebedev. Explicit difference schemes for solving stiff systems of ODEs and PDEs with complex spectrum. *Russian J. Numer. Anal. Math. Modelling*, 13(2):107–116, 1998.

[152] P. Lesaint. *Sur la résolution des sytèmes hyperboliques du premier ordre par des methodes déléments finis*. PhD thesis, University of Paris VI, 1975.

[153] P. Lesaint and P.-A. Raviart. On a finite element method for solving the neutron transport equation. pages 89–123. Publication No. 33, 1974.

[154] P. Lesaint and P. A. Raviart. On a finite element method for solving the neutron transport equation. *Publications mathmatiques et informatique de Rennes*, (S4):1–40, 1974.

[155] K. M. Levere. Changes in habitat of fish populations: An inverse problem. *Math. Biosci.*, 278:48–55, 2016.

[156] J. Liesen and Z. Strakoš. *Krylov subspace methods.* Numerical Mathematics and Scientific Computation. Oxford University Press, Oxford, 2013. Principles and analysis.

[157] S. Lipschutz, M. Spiegel, and D. Spellman. *Schaum's Outline of Vector Analysis, 2ed.* McGraw-Hill's ebook library: Student study aids. McGraw-Hill Education, 2009.

[158] W. Liu, M. Li, B. Lv, Y. Chen, H. Ma, A. S. Viana, J. P. Correia, and G. Jin. An imaging ellipsometry approach to dissolved oxygen measurement on surface tethered weak polyelectrolyte modified electrode. *Journal of The Electrochemical Society J. Electrochem. Soc.*, 163(5), 2016.

[159] J. D. Logan. *Transport modeling in hydrogeochemical systems*, volume 15 of *Interdisciplinary Applied Mathematics.* Springer-Verlag, New York, 2001. Geophysics and Planetary Sciences.

[160] F. Lörcher, G. Gassner, and C.-D. Munz. A discontinuous Galerkin scheme based on a space–time expansion. I. inviscid compressible flow in one space dimension. *Journal of Scientific Computing*, 32(2):175–199, 2007.

[161] F. Lörcher, G. Gassner, and C.-D. Munz. An explicit discontinuous Galerkin scheme with local time-stepping for general unsteady diffusion equations. *Journal of Computational Physics*, 227(11):5649 – 5670, 2008.

[162] C. Lucero. *Bayes risk A-optimal experimental design methods for ill-posed inverse problems.* ProQuest LLC, Ann Arbor, MI, 2013. Thesis (Ph.D.)–Colorado School of Mines.

[163] D. Macdonald. *Transient Techniques in Electrochemistry.* Springer US, 1977.

[164] T. P. Mathew, P. L. Polyakov, G. Russo, and J. Wang. Domain decomposition operator splittings for the solution of parabolic equations. *SIAM Journal on Scientific Computing*, 19(3):912–932, 1998.

[165] W. Menke. *Geophysical data analysis: discrete inverse theory.* Academic Press, San Diego, second edition, 1989.

[166] B. V. Minchev and W. M. Wright. A review of exponential integrators for first order semi-linear problems, 2005.

[167] J. Mitchell, J. B. W. Webber, and J. Strange. Nuclear magnetic resonance cryoporometry. *Physics Reports*, 461(1):1 – 36, 2008.

[168] C. Moler and C. Van Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Rev.*, 45(1):3–49 (electronic), 2003.

[169] C. Moler and C. Van Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Rev.*, 45(1):3–49 (electronic), 2003.

[170] P. M. Morse and H. Feshbach. *Methods of theoretical physics. 2 volumes.* McGraw-Hill Book Co., Inc., New York-Toronto-London, 1953.

[171] A. Neubauer. Tikhonov regularisation for nonlinear ill-posed problems: optimal convergence rates and finite-dimensional approximation. *Inverse Problems*, 5(4):541–557, 1989.

[172] J. Niesen and W. M. Wright. Algorithm 919: a Krylov subspace algorithm for evaluating the ϕ-functions appearing in exponential integrators. *ACM Trans. Math. Software*, 38(3):Art. 22, 19, 2012.

[173] J. Nitsche. Über ein variationsprinzip zur lösung von dirichlet-problemen bei verwendung von teilräumen, die keinen randbedingungen unterworfen sind. In *Abhandlungen aus dem mathematischen Seminar der Universität Hamburg*, volume 36, pages 9–15. Springer, 1971.

[174] J. Nitsche. On Dirichlet problems using subspaces with nearly zero boundary conditions. pages 603–627, 1972.

[175] J. Nocedal and S. J. Wright. *Numerical optimization*. Springer Series in Operations Research. Springer-Verlag, New York, 1999.

[176] J. Nocedal and S. J. Wright. *Numerical optimization*. Springer Series in Operations Research and Financial Engineering. Springer, New York, second edition, 2006.

[177] A. Ogata and R. Banks. *A Solution of the Differential Equation of Longitudinal Dispersion in Porous Media*. Fluid movement in earth materials. U.S. Government Printing Office, 1961.

[178] P. olof Persson, A. Edelman, R. R. Rosales, and P. olof Persson. *Mesh Generation for Implicit Geometries*. PhD thesis, Department of Mathematics, MIT, 2005.

[179] P. olof Persson and G. Strang. A simple mesh generator in matlab. *SIAM Review*, 46:2004, 2004.

[180] J. Peraire and P.-O. Persson. The compact discontinuous Galerkin (cdg) method for elliptic problems. *SIAM Journal on Scientific Computing*, 30(4):1806–1824, 2008.

[181] P.-O. Persson and G. Strang. A simple mesh generator in Matlab. *SIAM Rev.*, 46(2):329–345 (electronic), 2004.

[182] C. J. N. Philip B. Bedient, H. S Rifai. *Ground water contamination : transport and remediation*. Upper Saddle River, NJ : Prentice Hall, 2nd ed edition, 1999.

[183] F. Pioldi and E. Rizzi. A Full Dynamic Compound Inverse Method for output-only element-level system identification and input estimation from earthquake response signals. *Comput. Mech.*, 58(2):307–327, 2016.

[184] S. Piperno. Symplectic local time-stepping in non-dissipative dgtd methods applied to wave propagation problems. *M2AN Math. Model. Numer. Anal.*, 40(5):815–841 (2007), 2006.

[185] T. Poston and I. N. Stewart. *Taylor expansions and catastrophes*. Pitman, London-San Francisco, Calif.-Melbourne, 1976. Research Notes in Mathematics, Vol. 7.

[186] M. Price. *Introducing Groundwater*. Stanley Thornes, 1996.

[187] Z. Pu, R. Wang, J. Wu, H. Yu, K. Xu, and D. Li. A flexible electrochemical glucose sensor with composite nanostructured surface of the working electrode. *Sensors and Actuators: B. Chemical*, 230:801 – 809, 2016.

[188] J. N. Reddy. *An introduction to the finite element method*, volume 2. McGraw-Hill New York, 1993.

[189] W. H. Reed and T. R. Hill. Triangular mesh methods for the neutron transport equation. *J. Comput. Phys*, 131:73–479, 1973.

[190] J.-F. Remacle, X. Li, M. S. Shephard, and J. E. Flaherty. Anisotropic adaptive simulation of transient flows using discontinuous Galerkin methods. *Internat. J. Numer. Methods Engrg.*, 62(7):899–923, 2005.

[191] J.-F. Remacle, S. Soares Frazão, X. Li, and M. S. Shephard. An adaptive discretization of shallow-water equations based on discontinuous Galerkin methods. *Internat. J. Numer. Methods Fluids*, 52(8):903–923, 2006.

[192] V. Rene, J. Randin, and I. D. Raistrick. Effect of active surface area on the response time of electrochromic and electrolytic displays. *Journal of The Electrochemical Society J. Electrochem. Soc.*, 129(11):2451, 1982.

[193] J. R. Rice. Split Runge-Kutta method for simultaneous equations. *J. Res. Nat. Bur. Standards Sect. B*, 64B:151–170, 1960.

[194] M. Rietmann, D. Peter, O. Schenk, B. Uçar, and M. Grote. Load-balanced local time stepping for large-scale wave propagation. In *Parallel and Distributed Processing Symposium (IPDPS), 2015 IEEE International*, pages 925–935. IEEE, 2015.

[195] B. Riviere. *Discontinuous Galerkin methods for solving elliptic and parabolic equations: theory and implementation*. SIAM, 2008.

[196] B. Rivière, M. F. Wheeler, and V. Girault. Improved energy estimates for interior penalty, constrained and discontinuous Galerkin methods for elliptic problems. I. *Comput. Geosci.*, 3(3-4):337–360 (2000), 1999.

[197] Z. Ruan, J. Z. Yang, and X. Lu. Tikhonov regularisation method for simultaneous inversion of the source term and initial data in a time-fractional diffusion equation. *East Asian J. Appl. Math.*, 5(3):273–300, 2015.

[198] D. Schötzau. *hp-DGFEM for parabolic evolution problems: applications to diffusion and viscous incompressible fluid flow*. na, 1999.

[199] D. Schötzau and C. Schwab. An hp a priori error analysis of the DG time-stepping method for initial value problems. *Calcolo*, 37(4):207–232, 2000.

[200] P. K. Sekhar, J. Kysar, E. L. Brosha, and C. R. Kreller. Development and testing of an electrochemical methane sensor. *Sensors and Actuators B: Chemical*, 228:162 – 167, 2016.

[201] L. F. Shampine and M. W. Reichelt. The MATLAB ODE suite. *SIAM J. Sci. Comput.*, 18(1):1–22, 1997. Dedicated to C. William Gear on the occasion of his 60th birthday.

[202] L. F. Shampine, M. W. Reichelt, and J. A. Kierzenka. Solving index-1 DAEs in MATLAB and Simulink. *SIAM Rev.*, 41(3):538–552 (electronic), 1999.

[203] P. Sibanda and O. Makinde, editors. *A Mathematical Introduction to Incompressible Flow*. University of Zimbabwe Publications, Mount Pleant Harare, 2000.

[204] R. B. Sidje. A software package for computing matrix exponentials. *ACM Trans. Math. Software*, pages 130–156, 1998.

[205] R. D. Skeel and M. Berzins. A method for the spatial discretization of parabolic equations in one space variable. *SIAM J. Sci. Statist. Comput.*, 11(1):1–32, 1990.

[206] S. Skelboe and P. U. Andersen. Stability properties of backward Euler multirate formulas. *SIAM journal on scientific and statistical computing*, 10(5):1000–1009, 1989.

[207] G. Strang and G. J. Fix. *An analysis of the finite element method*, volume 212. Prentice-hall Englewood Cliffs, NJ, 1973.

[208] J. Sudirham, J. Van Der Vegt, and R. Van Damme. Space–time discontinuous Galerkin method for advection–diffusion problems on time-dependent domains. *Applied numerical mathematics*, 56(12):1491–1518, 2006.

[209] A. Tarantola. *Inverse problem theory*. Elsevier Science Publishers, B.V., Amsterdam, 1987. Methods for data fitting and model parameter estimation.

[210] A. Tarantola. *Inverse problem theory and methods for model parameter estimation*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2005.

[211] A. Taube, M. Dumbser, C.-D. Munz, and R. Schneider. A high-order discontinuous Galerkin method with time-accurate local time stepping for the Maxwell equations. *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields*, 22(1):77–103, 2009.

[212] V. A. Titarev and E. F. Toro. ADER: Arbitrary high order godunov approach. *Journal of Scientific Computing*, 17(1):609–618, 2002.

[213] J. Tromp, C. Tape, and Q. Liu. Seismic tomography, adjoint methods, time reversal and banana-doughnut kernels. *Geophys. J. Int*, 160:195216, 2005.

[214] P. Vabishchevich. Parallel domain decomposition algorithms for time-dependent problems of mathematical physics. *Advances in Numerical Methods and Applications*, pages 293–299, 1994.

[215] P. N. Vabishchevich. Two-component domain decomposition scheme with overlapping subdomains for parabolic equations. *Journal of Computational and Applied Mathematics*, 2017.

[216] P. J. van der Houwen and B. P. Sommeijer. On the internal stability of explicit, m-stage Runge-Kutta methods for large m-values. *Z. Angew. Math. Mech.*, 60(10):479–485, 1980.

[217] H. A. van der Vorst. *Iterative Krylov methods for large linear systems*, volume 13 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, Cambridge, 2009. Reprint of the 2003 original.

[218] R. A. Waltz, J. L. Morales, J. Nocedal, and D. Orban. An interior algorithm for nonlinear optimization that combines line search and trust region steps. *Math. Program.*, 107(3, Ser. A):391–408, 2006.

[219] T. C. Warburton and G. E. Karniadakis. A discontinuous Galerkin method for the viscous MHDequations. *J. Comput. Phys.*, 152(2):608–641, 1999.

[220] M. F. Wheeler. An elliptic collocation-finite element method with interior penalties. *SIAM J. Numer. Anal.*, 15(1):152–161, 1978.

[221] H. Yücel, M. Stoll, and P. Benner. A discontinuous Galerkin method for optimal control problems governed by a system of convection–diffusion pdes with nonlinear reaction terms. *Computers & Mathematics with Applications*, 70(10):2414–2431, 2015.

[222] V. Yurko. *Method of spectral mappings in the inverse problem theory*. Inverse and Ill-posed Problems Series. VSP, Utrecht, 2002.

[223] J. Zhao, H. Yin, T. Lim, H. Xie, H.-Y. Hsu, F. Forouzan, and A. J. Bard. Electrodeposition of photoactive silicon films for low-cost solar cells. *Journal of The Electrochemical Society J. Electrochem. Soc.*, 163(9), 2016.

[224] T. A. Zhou. Alkaline-pretreated aluminum electrodes as ph sensors. *Journal of The Electrochemical Society J. Electrochem. Soc.*, 141(5):1142, 1994.

[225] O. C. Zienkiewicz, R. L. Taylor, O. C. Zienkiewicz, and R. L. Taylor. *The finite element method*, volume 3. McGraw-hill London, 1977.

[226] C. Zoppou and J. Knight. Analytical solution of the spatially variable coefficient advective-diffusion equation in one , two, three dimensions. *Mathematics research report*, 1997.